Journal of Signal Processing Systems manuscript No.
(will be inserted by the editor)

On the Implementation of a Hardware Architecture for an Audio Data Hiding System

Jose Juan Garcia-Hernandez · Claudia Feregrino-Uribe · Rene Cumplido · Carolina Reta

Received: date / Accepted: date

Abstract Data hiding systems have emerged as a solution against the piracy problem, particularly those based on quantization have been widely used for its simplicity and high performance. Several data hiding applications, such as broadcasting monitoring and live performance watermarking, require a real-time multi-channel behavior. While Digital Signal Processors (DSP) have been used for implementing these schemes achieving real-time performance for audio signal processing, custom hardware architectures offer the possibility of fully exploiting the inherent parallelism of this type of algorithms for more demanding applications. This paper presents an efficient hardware implementation of a Rational Dither Modulation (RDM) algorithm-based data hiding system in the Modulated Complex Lapped Transform (MCLT) domain. In general terms, the proposed hardware architecture is conformed by an MCLT processor, an Inverse MCLT processor, a Coordinate Rotation Digital Computer (CORDIC) and an RDM-QIM processor. Results of implementing the proposed hardware architecture on a Field Programmable Gate Array (FPGA) are presented and discussed.

Keywords Data hiding · Audio Signal · FPGA · Multi-channel processing

1 Introduction

Expansion of the Internet service together with rapid advance of high capacity storage systems such as Compact Disc (CD) and Digital Versatile Disc (DVD) facilitated the fast and perfect copy of digital content. However, at the same time the use of these technologies cause serious problems, such as unauthorized copying and distribution of digital materials. Conventional cryptography systems encrypt digital data during its transmission and permit only authorized person to decrypt the encrypted data, however once such data are decrypted they are totally vulnerable to illegal copying and distribution. Digital watermarking (in this paper *data hiding* is used indistinctly) has been considered as a solution for these problems. During last decade several watermarking algorithms have been developed. Digital watermarking is a technique that embeds an imperceptible and statistically undetectable signal to the digital contents. Watermarking algorithms must satisfy some requirements, such as imperceptibility of embedded signal (watermark), robustness to some common intentional and non intentional attacks and high embedding data rate. Especially high performance audio watermarking algorithms are not easy to develop, because the human auditory system is more sensitive than human visual system and small changes to the audio signal due to the watermark embedding can be detected by human ears [1]. Additionally in audio watermarking systems blind watermark detection is required, because in many applications such as illegal copy control systems, distribution and broadcasting monitor system and audio steganography, original unwatermarked signal is not available in the detection stage.

In order to implement a real-time watermarking system it is possible to choose between two main platforms: Digital Signal Processors (DSP) and Field Programmable Gate Arrays (FPGA). Implementations on DSPs have been previously reported [22,23]. Those implementations do not exploit the possible parallelism of several watermarking algorithms. Technical outposts for DSP programming exist with the purpose of exploit-

All authors are with
The National Institute for the Astrophysics, Optics and Electronics, Puebla, Mexico
E-mail: {jjuan,cferegrino,rcumplido,creta}@ccc.inaoep.mx

ing the parallelism of algorithms, nevertheless multi-channel processing in demanding tasks, such as video processing, is not straightforward. FPGA-based implementation of data hiding systems seems to be an interesting option since its capacity for parallel processing could allow multi-channel processing.

Hardware implementations of data hiding systems have been poorly explored in the literature. In [24], the author reports an FPGA implementation of a video watermarking algorithm and its comparison with a DSP implementation. Implementation results for both FPGAs and DSP devices suggest that the FPGA is a better option in terms of processing speed, power consumption and device cost. A data hiding system for speech bandwidth and its hardware implementation is proposed in [25]. The system uses data hiding techniques to transmit high frequency speech components in order to improve the speech quality in transmission systems. The hardware implementation is carried out using application software and one FFT implemented in a hardware acceleration model. Due to the use of application software, the performance is limited in speed terms. In [26], a data hiding system using digital images as host signals and its hardware implementation is proposed. Performance results of the hardware implementation are superficially presented. However, the author claims that implementation using FPGA allows its application for real time multimedia data transmission. In [27], hardware implementations of steganographic techniques that can be applied to documents, images and video is reported. According to the authors, implementation results show that real time performance is guaranteed. In [28] an steganographic micro-architecture and its FPGA implementation is presented. The authors propose a video or audio steganographic model in which the hidden message can be composed and inserted in the cover medium in real time. Real time performance is demonstrated, with a reported throughput of 1.576 Mbps.

This work explores the suitability of exploiting the parallelism of an audio data hiding system in a hardware implementation. For prototyping and validation purposes, the full data hiding system architecture has been implemented in an FPGA. The outline of the paper is as follows: Section 2 presents a revision of state-of-the-art data hiding systems in audio signals. Section 3 details the proposed watermarking system. The circuit design, simulation results and hardware resources are presented in Section 4. System evaluation in terms of robustness and signal quality is presented in Section 5. Finally the conclusions are given in Section 6.

2 Data Hiding Systems in Audio Signals

Audio watermarking techniques can be classified in two groups: time domain techniques and frequency domain techniques. In the time domain techniques, watermark embedding is carried out directly in the audio signal, while in the frequency domain based system, the watermark signal is embedded in frequency domain, such as Discrete Fourier Transform (DFT), Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT). Generally it is difficult to satisfy two principal requirements (robustness and inaudibility), mentioned above, using the time domain watermarking systems, the main part of the psychoacoustic model is developed in the frequency domain. This is because embedding a watermark in the time domain in an imperceptible manner may be difficult mainly with non-linear data hiding algorithms [2]. Because of this it is necessary to transform the watermark from time domain to frequency domain, apply the psychoacoustic model and transform the watermark back to the time domain. Also time domain watermarking system can be vulnerable to some common attacks, such as MP3 compression, filtering, noise addition, etc [3].

During last decade, many audio watermarking algorithms have been proposed in literature [1,4–8], however no method can satisfy all requirements mentioned above. Echo hiding method is one of the successful temporal domain method [4–6], that embed binary bits using echo signal with different delays. Echo hiding method is usually imperceptible, however the method is vulnerable to some malicious attacks, such as echoing and in the watermark detection process requires high complexity computation [9]. The spread-spectrum watermarking method embeds a pseudo-random sequence generated by secret owner's key into some frequency bands of the audio signal in transformed-domain [7,10,11]. In this method, firstly audio signal is transformed by Discrete Cosine Transform (DCT), Discrete Fourier Transform (DFT), or Discrete Wavelet Transform (DWT), and a pseudo- random sequence is embedded in some frequency bands considering the imperceptibility and robustness requirements. Implementation of this method is generally simple, but robust embedding causes audible noise in the watermarked audio signal. A quantization scheme quantizes audio data using the determined quantizer and embedding watermark bit value. The Quantization Index Modulation (QIM) [12] appears to be a practical solution to the digital information hiding problem. The main task in QIM based method, such as Dither Modulation, is the design of suitable quantizers used to embed the data. However such simple method, as several other QIM based meth-

ods, presents lack of robustness against the gain attack, consisting in the multiplication of the host feature sequence by a gain factor p which is unknown to the decoder. In order to reduce that vulnerability, several schemes have been proposed [13,14]. Rational Dither Modulation (RDM) was introduced as a solution to gain attack in high-rate data hiding schemes [15].

On the other hand, when the watermarking process is carried out in a block transform domain like DCT, DFT or DWT the reconstructed signals exhibit the block artifact effect. In order to beat block artifacts, a family of lapped transforms was developed [16]; modulated lapped transform (MLT) is a member of that family, MLT uses *2M* samples in order to compute M coefficients. The MLT has been used in several audio coding standards [17]. However, MLT coefficients are only real, so, there is no phase information. In [18], the author proposed the Modulated Complex Lapped Transform (MCLT), which is an extension of MLT, but with complex components, also, fast MCLT algorithms based on discrete cosine transform and discrete sine transform were presented.

The MCLT domain has been satisfactorily used in audio watermarking [11,19,20], due to its no block artifact property [16]. In [21] the authors show that it is possible to hide about of 689 bits per second (bps) in a CD-quality audio signal, using RDM in MCLT domain.

3 Proposed System

Figures 1 and 2 show the proposed data hiding system, and the data recovery system, respectively. Original signal and watermarked signal are in Q15 format (along this paper we use aQb syntax, where a is the number of bits used to represent the integer part and b is the number of bits used to represent the fractional part). Each block in both figures is detailed in the following subsections.

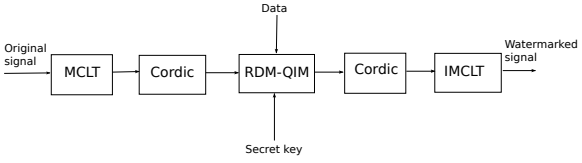

Fig. 1 Data hiding system

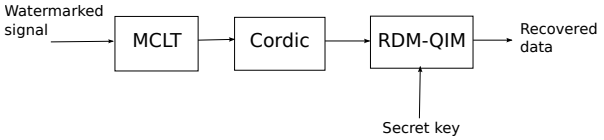

Fig. 2 Data recovery system

3.1 Malvar's Fast Algorithm for the MCLT/Inverse MCLT

In [29] the authors presented an FFT based fast algorithm and its CPLD implementation of the MCLT, however, that algorithm uses one pre-processing and one post-processing stage. Malvar showed in [30] that it is necessary only one post-processing stage after the FFT for the MCLT computing and one pre-processing stage before IFFT for the IMCLT computing.

3.1.1 Fast MCLT Algorithm

In [30] the author shows that the MCLT coefficients $X(k)$ can be obtained as follows:

$$X(k) = jV(k) + V(k+1) \tag{1}$$

where

$$\begin{aligned} V(k) &= c(k)U(k) \\ c(k) &= W_8(2k+1)W_{4M}(k) \\ U(k) &= \sqrt{\frac{1}{2M}} \sum_{n=0}^{2M-1} x(n)W_{2M}(kn) \end{aligned} \tag{2}$$

and

$$W_M(r) = \exp\left(\frac{-j2\pi r}{M}\right) \tag{3}$$

$U(k)$ is a $2M$ point FFT with orthonormal basis function of the input block $x(n)$, which means that MCLT coefficients can be computed by first computing FFT of $x(n)$ to obtain $U(k)$ and then to carry out the operations with factors $c(k)$.

3.1.2 Fast Inverse MCLT

To carry out the IMCLT in [30] the author developed the next relations:

$$Y(k) = \frac{c^*(k)}{4}[X(k-1) - jX(k)] \tag{4}$$

Where $X(k)$ are the MCLT coefficients, the superscript * denotes complex conjugation, and the modulation $c(k)$ is the same as that in (2). Using (4) we compute the M first FFT coeficientes of $y(n)$, but it is well known

that FFT coefficients must satisfy the conjugate symmetry property

$$Y(2M-k) = Y^*(k) \tag{5}$$

Finally, we know that $Y(0)$ and $Y(M)$ must be real-valued, after some manipulations,

$$\begin{aligned} Y(0) &= \frac{1}{\sqrt{8}}[\Re\{X(0)\} + \Im\{X(0)\}] \\ Y(M) &= -\frac{1}{\sqrt{8}}[\Re\{X(M-1)\} + \Im\{X(M-1)\}] \end{aligned} \tag{6}$$

with $\Re$ and $\Im$ taking the real and imaginary parts, respectively.

Malvar shows in [30] that equations (1), (4), (5) and (6), used with FFT processors were the fastest MCLT/IMCLT algorithms developed to that date. Next subsection shows the implementation of the equation (1) and FFT processor, corresponding to the MCLT processor, and the implementation of the equations (4), (5) and (6) and IFFT processor, corresponding to the inverse MCLT processor.

3.2 Rational Dither Modulation

In [12] it was proposed a class of data hiding methods called Quantization Index Modulation(QIM) and it works as follows: A scalar quantization scheme quantizes a vector of samples $\mathbf{x}$ and assigns a new value to the vector $\mathbf{x}$ based on the quantized vector value.

One of the worst attacks on QIM schemes is amplitude scaling. Rational Dither Modulation (RDM) was proposed as a possible solution to that attack by Perez-Gonzalez *et al.* [15], in order to get a high rate data hiding method invariant to gain attacks. The embedding rule is as follows:

$$y_k = g(\mathbf{y_{k-1}}) Q_{b_k}\left(\frac{x_k}{g(\mathbf{y_{k-1}})}\right) \tag{7}$$

where y_k is the RDM sample, $\mathbf{y_{k-1}}$ is a vector of $k-1$ past RDM samples, x_k is the host sample, Q_{b_k} is a message dependent quantizer and g is a function satisfying the property:

$$g(p\mathbf{y}) = pg(\mathbf{y}) \tag{8}$$

where p is the gain attack.

Decoding is carried out by following the expression:

$$b_k = arg\ min\left|\frac{z_k}{g(\mathbf{z_{k-1}})} - Q_{b_k}\left(\frac{z_k}{g(\mathbf{z_{k-1}})}\right)\right| \tag{9}$$

where b_k is the decoded bit, z_k is the received signal and $\mathbf{z_{k-1}}$ is a vector of $k-1$ past received signals.

The problem of choosing a particular g function is an important issue due to the intrinsic nonlinearity of the quantization process, Perez-Gonzalez *et al.* [15] suggests one subset based on Holder or l_p vector-norms:

$$g(\mathbf{y_{k-1}}) = \left(\frac{1}{L}\sum_{m=k-L}^{k-1} y_m^p\right)^{\frac{1}{p}}, p \geq 1 \tag{10}$$

where L is the number of past RDM samples utilized in the data hiding process. In [31] the authors proposed to use moving averages instead of function g. In this work the use of moving averages is also considered. Under that consideration the embedding rule becomes:

$$y_k = |\bar{y}_k| Q_{b_k}\left(\frac{x_k}{|\bar{y}_k|}\right) \tag{11}$$

and the detection is carried out like follows:

$$\hat{b}_k = argmin_{b_k=\{0,1\}}\left|z_k - |\bar{y}_k| Q_{b_k}\left(\frac{x_k}{|\bar{y}_k|}\right)\right| \tag{12}$$

$$Q\left(x_k, \bar{y}_k, v_{kb}\right) = q\left(\frac{x_k}{|\bar{y}_k|} + v_{kb}, \Delta\right) - v_{kb} \tag{13}$$

where $\bar{y_k}$ is the average of the 16 last y_k values and q is defined by equation 14

$$q(x, \Delta) = round\left(\frac{x}{\Delta}\right)\Delta \tag{14}$$

the v_{kb} values are generated as follows:

for $b = 0$ v_{kb} is a random value
for $b = 1$

$$\phi(v_{kb}) = \begin{cases} v_{k(b-1)} + \frac{\Delta}{2}; v_{(kb-1)} < 0 \\ v_{k(b-1)} - \frac{\Delta}{2}; v_{(kb-1)} \geq 0 \end{cases} \tag{15}$$

From the RDM-QIM algorithm it is possible to see that the pseudo-random numbers v_{kb} generation (equation 15) can be carried out at the same time that the rest of the procedures. Moreover, pseudo-random numbers can be generated using previously reported efficient hardware implementations of Linear Feedback Shift Registers (LFSR) [32]. In order to implement the function g (equation 10), which uses a memory block, it is possible to use a *register file*, which is able to perform several additions and accumulations in parallel form with the other modules of the algorithm. These characteristics make the RDM-QIM algorithm suitable for a compact and efficient hardware implementation in an FPGA.

4 FPGA Hardware Implementations

The architecture was modeled in VHDL and simulated using ModelSim. Synthesis results for the audio data hiding architecture are presented in this section. For the purpose of prototyping and validation, the architecture was synthesized, mapped, placed and routed for the Xilinx's Virtex-4 xc4vsx35 FPGA device using the Xilinx's ISE 9.1 design suite.

4.1 The MCLT and Inverse MCLT Processors[1]

The requirements for this MCLT implementation are: input data with format Q15, output data with format 9Q15 and $M = 128$. Figure 3 shows the direct MCLT processor. There are two blocks: an FFT processor and butterfly-like stage that performs equation 1. The FFT processor is implemented using a pre-designed core [34] configured in streaming mode.

The c factors are stored in a ROM using format Q15 in the butterfly-like stage, it also contains a register in order to store $V(k+1)$ when $X(k)$ is computed and the next clock cycle that value becomes $V(k)$. Figure 4 shows the butterfly-like structure, where *xk_re* and *xk_im* are the real and imaginary components of FFT output, *xk*, respectively, *xk_index* is the index of FFT value being processed, *c_re* and *c_im* are the real and imaginary components of factors c respectively, *V_re* and *V_im* are the real and imaginary components of V respectively and *sal_re* and *sal_im* are the real and imaginary components of *sal* MCLT coefficients respectively.

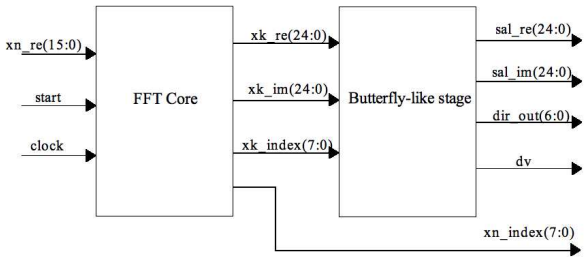

Fig. 3 Direct MCLT processor

When *start* goes high it begins the loading phase, input data $xn\_re(xn\_index)$ should arrive three cycles later than the xn_index it matches [34], therefore, it is possible to use input data from an external memory or a frame buffer. The MCLT processor was developed in streaming mode, so, after an initial latency of around 615 clock cycles, it begins outputting MCLT values $X(sal\_dir) = sal\_re(sal\_dir) + jsal\_im(sal\_dir)$ and *dv* goes high. There is a M clock cycles latency due to it is necessary to load $2M$ input samples in order to get M MCLT coefficients.

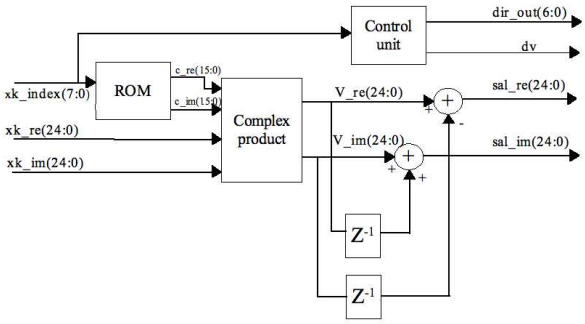

Fig. 4 Butterfly-like stage for the direct MCLT processor

The $X(sal\_dir)$ values are presented in 9Q15 format. The calculations carried out in the butterfly-like stage are 40 bit wide because c factors are in Q15 format and xk samples are in 9Q15 format, therefore, a product between a Q15 number and a 9Q15 number results in a 9Q30 number, so it is necessary to truncate to the most significative twenty five bits in order to satisfy the constraint previously imposed.

After *Place and Route* procedure the maximum clock rate is around 91 MHz. Due to the MCLT processor is designed in streaming mode and, after the initial latency, the MCLT processor gives a valid MCLT coefficient each clock cycle, it is possible to consider a length-128 MCLT computing in 2.8 μs. The performance demonstrated by our processor suggests it can be used for multi-channel applications, for example, in a typical block-based audio processing application, each 128 samples block is captured in 2.9 ms, if our MCLT processor is able to carry out a length-128 MCLT computing in 2.8 μs then it is possible to process around 1035 channels simultaneously. In a software implementation running on an Apple iMac, G5-based workstation with a 1.9 GHz processor and 2 GB of RAM [2] it was able to perform a length-128 MCLT computing in 625 μs. The system proposed in this paper performs around 220 times faster that this software implementation. For a multi-broadcasting monitoring application that performance is very useful. The processor presented in [29] is able to perform a length-16 MCLT in 6.06 μs, however, it is unfair to compare that implementation with our processor because the first one is implemented in a CPLD with smaller performance in comparison with the FPGA that we are using, but there

[1] These implementations were previously reported by the authors in [33]

[2] The same workstation is used for software implementatios along this work.

Table 1 FPGA's resources utilized for MCLT/IMCL implementations.

	Direct MCLT	Inverse MCLT
RAMB16s	7	14
Slices	2301	3545
BUFGMuxs	1	1
DSP48s	58	58
Max. Clock Frequency (MHz)	91.5	72.3
Throughput (MSPS)	91.5	72.3

are no more MCLT implementations using configurable structures reported in the literature.

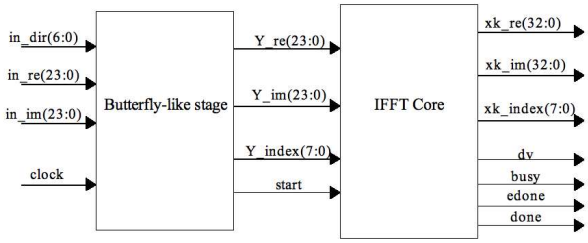

Fig. 5 Inverse MCLT processor

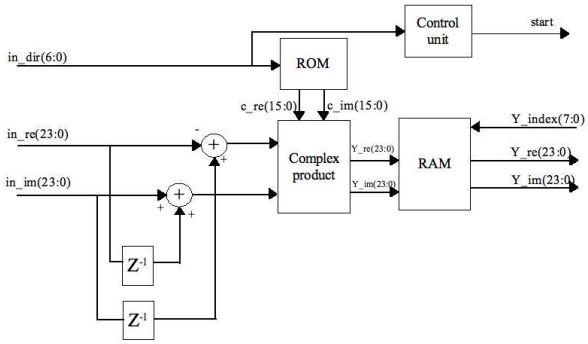

Fig. 6 Butterfly-like stage for the inverse MCLT processor

The inverse MCLT processor was implemented in a similar form, c^* factors are stored in a ROM in the butterfly-like stage block in figure 5. In this block, equations (4), (5) and (6) are computed.The MCLT coefficients are watermarked in a sequential form, therefore, only two watermarked coefficients, $in\_re(in\_dir) + jin\_im(in\_dir)$ and $in\_re(in\_dir-1) + jin\_im(in\_dir-1)$ are stored in a register system similar to the direct MCLT processor, however, it is necessary to store $Y(k)$ values in a RAM in order to keep them accessible to the IFFT core. Figure 6 shows that butterfly-like structure, where *in_re* and *in_im* are the real and imaginary components of the watermarked sample *in*, respectively, *in_index* is the index of watermarked sample being processed, *c_re* and *c_im* are the real and imaginary components of factors c^* respectively, *Y_re* and *Y_im* are the real and imaginary components of *Y*. Internal control signal, generated in the *control unit* block in figure 6, begins the loading process for IFFT core in the right-hand block in figure 5 and control signals of IFFT core indicate when inverse MCLT is done. The *busy* signal will go high when IFFT is being computed, *edone* goes high one clock cycle immediately after *done* goes active, *done* will transition high for one clock cycle when the transform calculation has completed, and finally, *dv* goes high when there is a valid value $xk\_re(xk\_index) + jxk\_im(xk\_index)$. After *Place and Route* procedure the maximum clock rate is around 72 MHz. Due to, again, the inverse MCLT processor is designed in streaming mode it is possible to consider a length-128 inverse MCLT computing in 3.5 μs. Table 1 shows the FPGA resources utilized for, both direct MCLT and inverse MCLT implementations, after *Place and Route* procedure. From table 1 it can be seen that the direct MCLT processor utilizes a minor number of slices and RAM16s components than the inverse MCLT processor does, it is due to the inverse MCLT processor uses a RAM stage and the direct MCLT processor does not. Moreover, the input samples for the inverse MCLT processor are 24 bits wide and, for the direct MCLT processor they are 16 bits wide, then a greater amount of slices for the inverse MCLT processor is necessary. The throughput is affected for the same width input conditions, in the direct MCLT processor it is 91.5 mega samples per second (MSPS) and for the inverse MCLT processor it is 72.3 MSPS. It is necessary to truncate the 16 least significant bits in order to keep the original signal width.

4.2 The Coordinate Rotation Digital Computer (CORDIC)

With the purpose of transforming complex MCLT magnitudes to polar representation and vice versa, a predesigned CORDIC core is used [35]. In order to transform from rectangular to polar the CORDIC is configured as rotate vector and, in the other hand, in order to transform from polar to rectangular the CORDIC

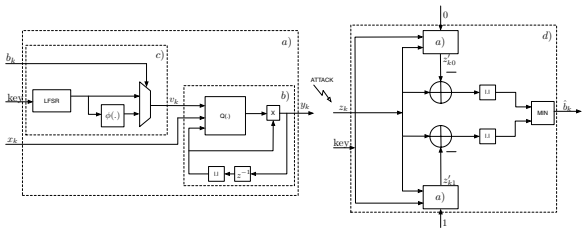

Fig. 7 Algorithm RDM-QIM. a) Insertion block diagram, b) Quatifier block diagram, c) The v_k generation stage, d) Detection block diagram.

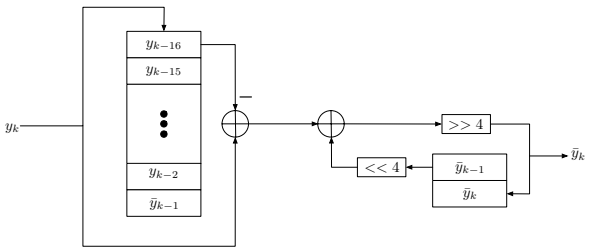

Fig. 8 Architecture for the function $g(.)$ computing

is configured as translate vector. Table 2 shows the FPGA's resources utilized for CORDIC implementations after *Place and Route* procedure. Both implementations are carried out with 25 iterations.

4.3 The RDM-QIM Algorithm[3]

Figure 7 shows a block diagram of the algorithm RDM-QIM.

The insertion block (figure 7 module a)) allows concealing a message's bit b_k within a carrier signal x_k using a *key* to obtain an output signal y_k extremely similar to the input signal x_k. The signal y_k can be altered by an attack and be transformed into z_k. The detection block is able to retrieve the inserted bit on the signal z_k through the estimation of $\hat{b}_k$ using the same *key*. The hardware design of the insertion block, based on equation 11, is composed of blocks in figure 7 module b) and 7 module c)

The quantifier Q (figure 7 module b)) used by the algorithm represents equation 13 and requires a value v_k generated by the *key* and inserted bit b_k, as well as the carrier signal and the reference value which represents the past events of the output signal.

The v_k generation stage (figure 7 module c)) represents equation 15. The Linear Feedback Shift Register (LFSR) block can generate pseudo-random numbers in Q8 format from an specified key using a shift register and taps according to the LFSR polinomy $x^6 + 1$ which defines the largest sequence for a Q8 format [37]. The transformation ϕ is implemented according to the equation 15 using Q8 format and $\Delta = 0.25$. Depending on the bit b_k, the value v_k can be the pseudo-random number generated by LFSR or the value of the transformation ϕ. The algorithm requires a representative value of past events, therefore a memory is needed. The memory stores information in 17Q8 format of the last 16 frames, so that when a new value arrives the old value is replaced in the corresponding position and frame. The representative value is obtained by averaging the values stored in the specific position of the 16 frames (function $g(.)$). It is important to highlight that to avoid computing the average by accessing 16 times the memory, an auxiliary memory stores the reference values in 17Q8 format of each frame and updates these values using the equation 16,

$$\bar{y}_k = \frac{\bar{y}_{k-1} * 16 + y_k - y_{k-16}}{16} \tag{16}$$

where $\bar{y}_{k-1}$ is the preceding average, y_k is the current output value and y_{k-16} is the output value of the 16th event. Both memories are initialized with 1s. By computing the average in this way, fourteen adding operations are avoided at the cost of an extra shift operation. Figure 8 shows the implementation of the average computing using the equation 16. It has been shown that it provides higher robustness to hide a symbol using several samples instead of one [21]. Modules as the one shown in figure 8 allow to exploit the intrinsic parallelism of FPGA devices in block-based data hiding systems, for example, if 64 samples are used to hide one bit, it is possible to generate 64 modules working in parallel fashion.

The division operations are performed by using the division core from Xilinx ISE configured in pipeline with latency of 54 cycles to accept dividends and divisors of 25 bits obtaining both quotient and residue of 25 bits too. The division result in format 24Q24 is compacted to generate a value with 17Q8 format by truncating the 7 MSBs and the 16 LSBs. Due to the latency generated by the division, a control unit is needed to generate the control signals for the algorithm and to store the input data in a memory avoiding multiple delays in the signals. The hardware design of the detection block is based on equation 12 and it is shown in figure 7d).

In the detection stage the quantization Q is done for both values $b_k = \{0, 1\}$. The quantization results multiplied by the reference value generate z'_{k0} and z'_{k1}. It is important to mention that at this stage the reference value is determined by the input value z_k. Except

[3] This implementation was previously reported by the authors in [36]

Table 2 FPGA's resources utilized for CORDIC implementations.

	Rotate vector	Translate vector
Slices	1254	1412
BUFGMuxs	1	1
DSP48s	4	8
Max. Clock Frequency (MHz)	254	254

for this, all the blocks designed for the insertion stage are used in the same way. After *Place and Route* procedure the maximum clock rate for the insertion and detection stage is 84.8 MHz and 60.2 MHz respectively. One sample is processed at each clock cycle in both stages, therefore, the throughput for the insertion and detection stages is 84.8 and 60.2 MSPS, respectively. In a DSP implementation using a TMS320C6416 device it is possible to achieve a throughput for the insertion and detection stage of 690 and 440 kilo samples per second (KSPS). It is important to note that our FPGA-based RDM-QIM implementation overcomes the DSP implementation for more than two orders of magnitude which is higher than the average improvement for DSP applications [38,39].

4.3.1 RDM-QIM Implementation Results

Table 3 shows the FPGA's resources utilized for RDM-QIM implementations after *Place and Route* procedure. From table 3 it can be seen that the clock rate of detection stage is the slower in the whole data recovery system, therefore, the maximum clock rate of the data recovery system will be about 60 MHz. On the other hand, clock rate of insertion stage does not influence in maximum clock rate of the whole data hiding system, because in that system the slower block is the IMCLT processor.

Finally, table 4 shows the FPGA's resources utilized for the proposed watermarking system implementation after *Place and Route* procedure.

From table 4 it can be seen that in a real-time multi-channel watermarking fashion it is possible to process, for embedding, 819 channels and for detection, 682 channels of CD-quality audio signals due to each sample is processed in a clock cycle. In a software implementation it was able to process around of 5.1 channels simultaneously. The system proposed in this paper performs around 160 times faster that a software implementation.

5 System Evaluation

This section presents the robustness and quality signal proofs applied to the watermarked signal. Due to the hardware implementation was carried out using fixed point arithmetic, there could be slight variations with respect to the software implementation that uses floating point arithmetic. Therefore, it is interesting to compare the robustness and quality signal results of the hardware and software implementations.

5.1 Robustness

In order to evaluate the robustness, classical audio watermarking attacks were applied to the watermarked signal. Table 5 shows the bit error rate (BER) of extracted hidden data after the Audio Stirmark attacks [40] are applied. Generally, the embedded data are robust to various types of attacks, except copysample, cutsample, echo, ffttests, flippsample and voicemove attacks that are not so important since they considerably distort the audio signal. From table 5 it is possible to observe that the hardware implementation and software implementation results are very close. On the other hand, BER showed in table 5 are very close to the DSP implementation results reported in [21].

5.2 Imperceptibly Proof

In order to evaluate the signal quality after watermarking process, the Modified Bark Spectral Distortion (MBSD) prove was carried out [41,42]. The MBSD measure estimates speech distortion in loudness domain taking into account the noise-masking threshold in order to include only audible distortions in the calculation of the distortion measure. That proof was carried out using 5 different kinds of music: Classic music, Rock music, Pop music, Instrumental music and Latin music. Results shown in table 6 suggest that the watermark is transparent to the human auditory system. Software implementation presents slightly better performance that the hardware implementation. However, this difference is negligible.

6 Conclusions

In this paper, a hardware implementation of a proposed audio data hiding system is presented. Using an FPGA

Table 3 FPGA's resources utilized for RDM-QIM implementations.

	Insertion stage	Detection stage
RAMB16s	5	5
Slices	1684	1784
BUFGMuxs	1	1
DSP48s	3	6
Max. Clock Frequency (MHz)	84.8	60.2

Table 4 FPGA's resources utilized for the proposed watermarking system implementation.

	Embedding system	Detection system
RAMB16s	26	12
Slices	10196	5303
BUFGMuxs	1	1
DSP48s	131	68
Max. Clock Frequency (MHz)	72.3	60.2

Table 5 Bit Error Rate Results on Audio Signals Attacked With the Stirmark Audio Benchmark. Hardware (HW) and software (SW) results.

	BER			BER			BER	
Attack	HW	SW	Attack	HW	SW	Attack	HW	SW
Original	0.000	0.000	Addbrum 100	0.000	0.000	Addbrum 1100	0.000	0.000
Addbrum 3100	0.000	0.000	Addbrum 4100	0.000	0.000	Addbrum 5100	0.000	0.000
Addbrum 7100	0.000	0.000	Addbrum 8100	0.000	0.000	Addbrum 9100	0.000	0.000
Addfftnoise	0.120	0.110	Addnoise 100	0.000	0.000	Addnoise 300	0.060	0.058
Addnoise 700	0.140	0.150	Addnoise 900	0.210	0.200	Addsinus	0.000	0.000
Compressor	0.220	0.200	Copysample	0.820	0.810	Cutsamples	0.800	0.800
Echo	0.610	0.610	Exchange	0.250	0.250	Extraestereo 30	0.180	0.160
Extraestereo 70	0.340	0.340	Fft hlpass	0.050	0.030	Fft invert	0.020	0.020
Fft stat	0.100	0.100	Fft test	0.900	0.900	Flippsample	0.810	0.800
Lsbzero	0.000	0.000	Normalize	0.000	0.000	Nothing	0.000	0.000
Lowpass	0.010	0.010	Resampling	0.000	0.000	Smooth	0.110	0.110
Stat1	0.090	0.000	Stat2	0.080	0.060	Voiceremove	0.780	0.780
Addbrum 2100	0.000	0.000	Addbrum 6100	0.000	0.000	Addbrum 10100	0.000	0.000
Addnoise 500	0.120	0.120	Amplify	0.000	0.000	Dynnoise	0.200	0.200
Extraestereo 50	0.230	0.210	Fft real reverse	0.010	0.010	Invert	0.000	0.000
Highpass	0.000	0.000	Smoth2	0.120	0.120	Zerocross	0.260	0.204

Table 6 MBSD evaluation results for five different kinds of music. Hardware (HW) and software (SW) results.

	MBSD (dB)	
Music kind	HW	SW
Classic music	-63.0	-64.0
Rock music	-65.1	-66.0
Pop music	-62.4	-63.0
Instrumental music	-61.3	-62.0
Latin music	-62.0	-62.0

it has been demonstrated that the system is able to process, for embedding, 819 channels and for detection, 682 channels of CD-quality audio signals. The computing time for each system suggests that in a watermarking-based multi-broadcasting application our implementations will be very adequate. On the other hand, although our MCLT/IMCLT implementations are part of a data hiding system these can be used in other different digital signal processing tasks such as noise cancellation and acoustic echo cancellation with same precision requirements. In the same way, due to the RDM-QIM proposed architecture's compact footprint, it can be used as accelerator in microprocessor-based systems for embedded applications or as a core in custom architectures. Robustness and quality signal proofs

shown that watermarked signal is resistant to the main attacks and transparent to the human auditory system.

Acknowledgment

The authors would like to thank CONACyT for financial support.

References

1. M. Arnold, "Audio watermarking: Features, applications and algorithms," in *IEEE International Conference on Multimedia and Expo*, vol. 2, 2002, pp. 49–54.
2. R. A. Garcia, "Digital watermarking of audio signals using psychoacoustic auditory model and spread spectrum theory," Master's thesis, School of Music, University of Miami, 1999.
3. J. J. Garcia-Hernandez, M. Nakano, and H. Perez, "Real time audio watermarking," *Journal of Telecommunications and Radio Engineering ISSN 0040-2508*, vol. 65, no. 4, pp. 327–340, 2006.
4. B. S. Ko, R. Nishimura, and Y. Suzuki, "Time spread echo method for digital audio watermarking," *IEEE Trans. on Multimedia*, vol. 7, no. 2, pp. 212–221, 2005.
5. H. O. Oh, J. W. Seok, J. W. Hong, and D. H. Youn, "New echo embedding technique for robust and imperceptible audio watermarking," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 3, 2001, pp. 1341–1344.
6. H. J. Kim and Y. H. Choi, "A novel echo hiding algorithm," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 13, no. 8, pp. 885–889, 2003.
7. J. Seok, J. Hong, and J. Kim, "A novel audio watermarking algorithm for copyright protection of digital audio," *ETRI Journal*, vol. 24, pp. 181–189, 2002.
8. I.-K. Yeo and H. J. Kim, "Modified patchwork algorithm: A novel audio watermarking scheme," *IEEE Trans. on Speech and Audio Processing*, vol. 11, no. 4, 2003.
9. F. Petitcolas, R. Anderson, and M. kuhn, "Attacks on copyright marking systems," *Lecture Note in Computer Science*, vol. 1525, pp. 218–238, 2001.
10. I. Cox, M. Miller, and J.Bloom, *Digital Watermarking*. Morgan Kaufmann Publisher, 2003.
11. D. Kirovski and H. Malvar, "Spread spectrum watermarking of audio signals," *IEEE Trans. on Signal Processing*, vol. 51, no. 4, pp. 1020–1033, April 2003.
12. B. Chen and G. Wornell, "Quantization index modulation: a class of provably good methods for digital watermarking and information embedding," *IEEE Trans. on Information Theory*, vol. 47, no. 4, pp. 1423–1443, May 2001.
13. J. Eggers, R. Bauml, and B. Girod, "Estimation of amplitude modifications before scs watermark detection," in *Security and Watermarking of Multimedia Contents IV, Proc. SPIE*, vol. 4675, 2002, pp. 387–398.
14. K. Lee, D. Kim, and K.A.Moon, "Em estimation of scale factor for quantization-based audio watermarking," in *Proceedings of Second International Workshop on Digital Watermarking*. Springer Verlag, October 2003, pp. 316–327.
15. F. P. Gonzalez, C. Mosquera, M. Barni, and A. Abrardo, "Rational dither modulation: a high rate data-hiding method invariant to gain attacks," *IEEE Trans. on Signal Processing*, vol. 53, pp. 3960–3975, October 2005.
16. H. S. Malvar, *Signal Processing with Lapped Transforms*. Artech House, Inc., 1992.
17. S. Shlien, "The modulated lapped transform, its time-varying forms, and its applications to audio coding standards," *IEEE Trans. on Speech and Audio Processing*, vol. 5, pp. 359–366, 1997.
18. H. S. Malvar, "A modulated complex lapped transform and its applications to audio processing," Microsoft Research, Tech. Rep., 1999.
19. D. Kirovski and H. Malvar, "Robust covert communication over a public audio channel using spread spectrum," in *4th International Information Hiding Workshop*, April 2001.
20. R. Zezula and J. Misurec, "Audio signal watermarking in mclt domain with the aid of 2d pattern," in *Proceedings of 2nd International Conference on Digital Telecommunications, ICDT '07*, 2007.
21. J. J. Garcia-Hernandez, M. Nakano, and H. Perez, "Data hiding in audio signals using rational dither modulation," *IEICE Electron. Express, ISSN 1405-5546*, vol. 5, no. 7, pp. 217–222, 2008.
22. ——, "Real time implementation of low complexity audio watermarking algorithm," in *Proceedings of the Third International Workshop on Random Fields and Processing in Inhomogeneous Media*, Guanajuato, Mexico, October 2005.
23. ——, "Real time mclt audio watermarking and comparison of several whitening methods in receptor side," *Computacion y Sistemas Journal ISSN 1405-5546*, vol. 11, no. 1, pp. 61–75, 2007.
24. W. Irizarry-Cruz, "Fpga implementation of a video watermarking algorithm," Master's thesis, University of Puerto Rico, Mayaguez Campus, 2006.
25. F. Wu, S. Chen, and H. Leung, "Data hiding for speech bandwidth extension and its hardware implementation," in *2006 IEEE International Conference on Multimedia and Expo*, 2006, pp. 1277–1280.
26. S. Mainty, A. Banerjee, and M. Kundu, "An image-in-image communication scheme and vlsi implementation using fpga," in *IEEE Indian annual conference (INDICON 2004)*, 2004, pp. 6–11.
27. H. Y. Leung, L. M. Cheng, L. L. Cheng, and C. Chan, "Hardware realization of steganographic techniques," in *Third International Conference on International Information Hiding and Multimedia Signal Processing (IIH-MSP 2007)*, vol. 1, 2007, pp. 279–282.
28. M. Saeb and H. Farouk, "Design and implementation of a secret key steganographic micro-architecture employing fpga," in *The Conference on Design, Automation and Test in Europe*, I. C. Society, Ed., vol. 3, 2004.
29. H.-M. Tai and C. Jing, "Design and efficient implementation of a modulated complex lapped transform processor using pipelining technique," *IEICE Trans. Fundamentals*, vol. E84-A, no. 5, pp. 1280–1286, May 2001.
30. H. S. Malvar, "Fast algorithm for the modulated complex lapped transform," Microsoft Research, Tech. Rep., 2005.
31. J. Oostven, T. Kalker, and M. Staring, "Adaptive quantization watermarking," in *Security, Steganography and Watermarking of Multimedia Contents VI Proc. SPIE*, vol. 5306, 2004, pp. 296–303.
32. J. F. Wakerly, *Digital Design Principles and Practices*. Prentice Hall, 2006.
33. J. J. Garcia-Hernandez, C. Feregrino-Uribe, and R. Cumplido, "Fpga implementation of a modulated complex lapped transform for watermarking systems," in *Proceedings of Reconfig 08, Cancun, Mexico.*, I. C. Society, Ed., 2008, pp. 367–372.
34. X. Inc., *Fast Fourier Transform v4.1*, Xilinx. Inc., April 2007.
35. ——, *Cordic V3.0*, Xilinx. Inc., May 2004.

36. J. J. Garcia-Hernandez, C. Reta, R. Cumplido, and C. Feregrino-Uribe, "Efficient implementation of the rdm-qim algorithm in an fpga," *IEICE Electron. Express, ISSN 1405-5546*, vol. 6, no. 14, pp. 1064–1070, 2009.
37. J. G. Proakis, *Digital Communications*. McGraw Hill, 1983.
38. I. Berkeley Design Technology, "Enabling technologies for sdr: Comparing fpga and dsp performance," Presented at SDR Conference, November 2006.
39. ——, "Comparing fpgas and dsps for high-performance dsp applications," Presented at GSPx Conference, November 2006.
40. M. Steinebach, J. Dittmann, C. Seibel, L. C. Ferri, F. A. Petitcolas, N. Fates, C. Fontaine, and F. Raynal, "Stirmark benchmark: Audio watermarking attacks," in *International Conference on Information Technology: Coding and Computing (ITCC '01)*, 2001.
41. W. Yang, M. Dixo, and R. Yantorno, "A modified bark spectral distortion measure which uses noise masking threshold," in *IEEE Speech Coding Workshop*, 1997, pp. 55–56.
42. W. Yang, M. Benbouchta, and R. Yantorno, "Performance of the modified bark spectral distortion as an objective speech quality measure," in *ICASSP*, vol. 1, 1998, pp. 541–544.