IEICE Electronics Express, Vol.6, No.14, 1064–1070

Efficient implementation of the RDM-QIM algorithm in an FPGA

Jose Juan Garcia-Hernandez[a)], **Carolina Reta**[b)], **Rene Cumplido**[c)], **and Claudia Feregrino-Uribe**[d)]

National Institute for the Astrophysics, Optics and Electronics (INAOE), Puebla, Mexico

a) *jjuan@ccc.inaoep.mx*

b) *creta@ccc.inaoep.mx*

c) *rcumplido@ccc.inaoep.mx*

d) *cferegrino@ccc.inaoep.mx*

Abstract: The RDM-QIM algorithm has been proposed as a solution to the gain attack in QIM-based data hiding schemes. Several data hiding applications, such as broadcasting monitoring and live performance watermarking, requires 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, FPGAs offer the posibility of fully exploiting the inherent parallelism of this type of algorithms for more demanding applications. This letter presents an efficient FPGA implementation of RDM-QIM algorithm that overcomes a DSP-based implementation for more than two orders of magnitude and allows real-time multi-channel behavior.

Keywords: data hiding, rational dither modulation, hardware architectures, FPGA

Classification: Science and engineering for electronics

References

[1] B. Chen and G. Wornell, "Quantization index modulation: a class of provably good methods for digital watermarking and information embedding," *IEEE Trans. Inf. Theory*, vol. 47, pp. 1423–1443, May 2001.

[2] 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, pp. 387–398, 2002.

[3] K. Lee, D. Kim, and K. A. Moon, "Em estimation of scale factor for quantization-based audio watermarking," in *Proc. of Second Int. Workshop on Digital Watermarking*, Springer Verlag, pp. 316–327, Oct. 2003.

[4] 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. Signal Process.*, vol. 53, pp. 3960–3975, Oct. 2005.

[5] J. J. Garcia-Hernandez, M. Nakano, and H. Perez, "Real time implementation of low complexity audio watermarking algorithm," in *Proc. of the Third Int. Workshop on Random Fields and Processing in Inhomogeneous Media*, (Guanajuato, Mexico), Oct. 2005.

DOI: 10.1587/elex.6.1064
Received June 11, 2009
Accepted June 24, 2009
Published July 25, 2009

[6] J. J. Garcia-Hernandez, M. Nakano, and H. Perez, "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.
[7] J. Oostven, T. Kalker, and M. Staring, "Adaptive quantization watermarking," in *Security, Steganography and Watermarking of Multimedia Contents VI Proc. SPIE*, vol. 5306, pp. 296–303, 2004.
[8] J. G. Proakis, *Digital Communications*. McGraw Hill, 1983.
[9] 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.
[10] I. Berkeley Design Technology, "Enabling technologies for sdr: Comparing fpga and dsp performance," *Presented at SDR Conference*, Nov. 2006.
[11] I. Berkeley Design Technology, "Comparing fpgas and dsps for high-performance dsp applications." *Presented at GSPx Conference*, Nov. 2006.

1 Introduction

Data hiding methods based on Quantization Index Modulation (QIM) [1] have shown to be a practical solution to the secret communications problem. The QIM algorithm can be applied to media data such as audio, images and video. The main task in the QIM method is to produce the codebooks that the quantizers use to embed the data. Some simple solutions like Dither Modulation are widely used to carry out that task. However, the main weakness of QIM is its debility against the gain attack. In order to beat that debility several schemes have been proposed [2, 3, 4]. Rational Dither Modulation (RDM) was introduced as a solution to the gain attack in high-rate data hiding schemes [4]. The basic idea behind RDM is to apply conventional Dither Modulation to the ratio of the current host sample and the previously generated watermarked sample. The performance can be improved by watermarking the ratio between the current feature sample and a proper function of the L previously watermarked samples. In applications such as broadcasting monitoring and live performance watermarking it is necessary to use a real-time multi-channel data hiding system. In order to implement a real-time data hiding system it is possible to choose between two main platforms: Digital Signal Processors (DSP) and Field Programmable Gate Arrays (FPGA). Some implementations on DSPs have been previously reported [5, 6], however, they do not exploit the possible parallelism of several data hiding algorithms. Technical outposts for DSP programming exist with the purpose of exploiting 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. To the best of our knowledge there is no RDM-QIM implementation in FPGA reported in the literature. In this letter an efficient FPGA

DOI: 10.1587/elex.6.1064
Received June 11, 2009
Accepted June 24, 2009
Published July 25, 2009

implementation of RDM-QIM algorithm is presented wich allows real-time multi-channel behavior.

2 Quantization Index Modulation

In reference [1], a class of data hiding methods called Quantization Index Modulation (QIM) was proposed, where 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. This scheme is formulated as follows:

$$\mathbf{y_m} = q(\mathbf{x} + \mathbf{d_m}, D) - \mathbf{d_m} \tag{1}$$

where $\mathbf{y_m}$ is the quantized vector, $\mathbf{x}$ is the host signal vector, $\mathbf{d_m}$ corresponds to dither vector, m is the symbol to be hiden (0 or 1), and $q(.)$ is a quantization function with quantization step D. $q(.)$ is given by

$$q(\mathbf{x}, D) = D\ round\left(\frac{\mathbf{x}}{D}\right) \tag{2}$$

where $round(\mathbf{x})$ rounds the elements of $\mathbf{x}$ to the nearest integers and the $\mathbf{d_m}$ vectors are generated as follows:

for $m = 0$ $\mathbf{d_m}$ is a random vector
for $m = 1$
$\mathbf{d_m} = \mathbf{d_{m-1}} + D/2$ if $\mathbf{d_{m-1}} < 0$
$\mathbf{d_m} = \mathbf{d_{m-1}} - D/2$ otherwise

2.1 Rational Dither Modulation

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.* [4], in order to get a high rate data hiding method invariant to gain attacks. The embedding rule is as follows:

$$y_k = g(\mathbf{y_{k-1}})Q_{b_k}\left(\frac{x_k}{g(\mathbf{y_{k-1}})}\right) \tag{3}$$

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:

$$g(p\mathbf{y}) = pg(\mathbf{y}) \tag{4}$$

where p is the gain attack.
Decoding is carried out by following the expression:

$$b_k = arg\ min\left|\frac{z_k}{g(\mathbf{z_{k-1}})} - Q_{b_k}\left(\frac{z_k}{g(\mathbf{z_{k-1}})}\right)\right| \tag{5}$$

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.

DOI: 10.1587/elex.6.1064
Received June 11, 2009
Accepted June 24, 2009
Published July 25, 2009

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.* [4] suggests one subset based on Holder or l_p vector-norms:

$$g(\mathbf{y_{k-1}}) = \left(\frac{1}{L}\sum_{m=k-L}^{k-1} y_m^p\right)^{\frac{1}{p}}, p \geq 1 \tag{6}$$

where L is the number of past RDM samples utilized in the data hiding process. In [7] the authors proposed to use moving averages instead of function g, in this letter that proposition is considered. Under that consideration the embedding rule becomes:

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

and the detection is carried out like follows:

$$\hat{b}_k = argmin_{b_k=\{0,1\}}\left|z_k - |y_{k-1}|Q_{b_k}\left(\frac{x_k}{|y_{k-1}|}\right)\right| \tag{8}$$

$$Q\left(x_k, y_{k-1}, v_{kb}\right)) = q\left(\frac{x_k}{|y_{k-1}|} + v_{kb}, \Delta\right) - v_{kb} \tag{9}$$

where q is defined by equation 10

$$q(x, \Delta) = round\left(\frac{x}{\Delta}\right)\Delta \tag{10}$$

the v_{kb} values are generated as follows:

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

$$\phi(v_{kb}) = \begin{cases} v_{k(b-1)} + \frac{\Delta}{2}; v_{(kb-1)} < 0 \\ v_{k(b-1)} - \frac{\Delta}{2}; v_{(kb-1)} \geq 0 \end{cases} \tag{11}$$

From the RDM-QIM algorithm it is possible to see that the pseudo-random numbers v_{kb} generation (equation 11) can be carried out at the same time that the rest of the procedures. Moreover, pseudo-random numbers can be generated using a Linear Feedback Shift Register (LFSR) from some efficient hardware implementation previously reported in the literature. In order to implement the function g (equation 6), 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. The next section presents the RDM-QIM hardware implementation.

3 FPGA implementation

Figure 1 shows a block diagram of the algorithm RDM-QIM.

The insertion block (figure 1 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

DOI: 10.1587/elex.6.1064
Received June 11, 2009
Accepted June 24, 2009
Published July 25, 2009

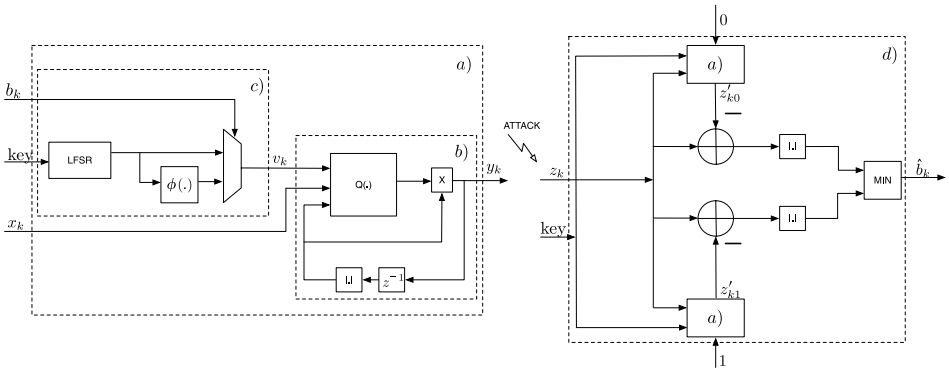

Fig. 1. Algorithm RDM-QIM. a) Insertion block diagram, b) Quatifier block diagram, c) The v_k generation stage, d) Detection block diagram.

Table I. FPGA's resources utilized for RDM-QIM implementation.

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

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 7, is composed of blocks in figure 1 module b) and 1 module c)

The quantifier Q (figure 1 module b)) used by the algorithm represents equation 9 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 1 module c)) represents equation 11. The Linear Feedback Shift Register (LFSR) block can generate pseudo-random numbers in Q8 (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) 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 [8]. The transformation ϕ is implemented according to the equation 11 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 necessary. 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

DOI: 10.1587/elex.6.1064
Received June 11, 2009
Accepted June 24, 2009
Published July 25, 2009

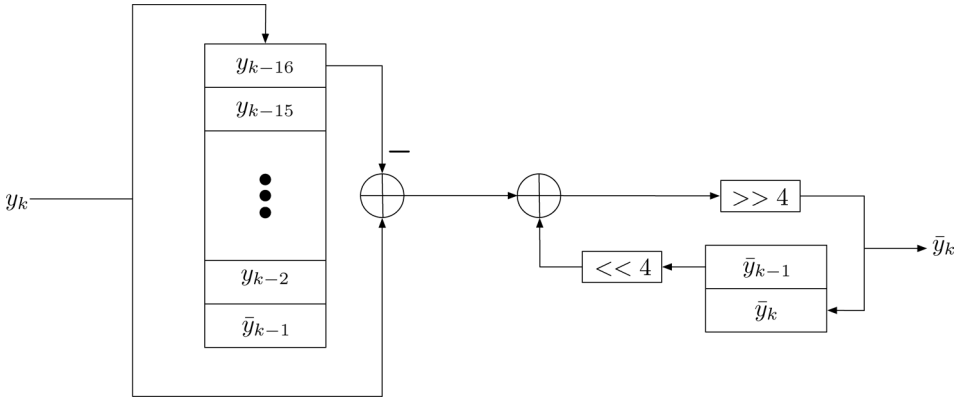

Fig. 2. Architecture for the function $g(.)$ computing.

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 auxiliar memory stores the reference values in 17Q8 format of each frame and updates these values using the equation 12,

$$\bar{y}_k = \frac{\bar{y}_{k-1} * 16 + y_k - y_{k-16}}{16} \tag{12}$$

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 2 shows the implementation of the average computing using the equation 12. It has been shown that to hide a symbol using several samples instead one provides higher robustness than one sample-based systems [9]. Modules as the one shown in figure 2 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 MSB's and the 16 LSB's. Due to the latency generated by the division, a control unit is necessary 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 8 and it is shown in figure 1d).

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 for this, all the blocks designed for the insertion stage are used in the same way. For simulation purposes, the insertion and detection stages were implemented in a Virtex-4 xc4vsx35 FPGA. After *Place and Route* procedure the maximum clock rate for the insertion and detection stage is 84.8 MHz and 60.2 MHz respec-

DOI: 10.1587/elex.6.1064
Received June 11, 2009
Accepted June 24, 2009
Published July 25, 2009

tively. 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 mega samples per second (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 k 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 [10, 11]. The high throughput of our RDM-QIM hardware implementation shows that it can be used in real-time multi-channel applications, for example: it can process 245.3 images (720×480) per second embedding a bit in every pixel, and recover hidden data in 174.1 images per second of the same quality which is enough for real-time multi-channel data hiding in video signals. To the best of our knowledge there is no other RDM-QIM implementation reported in the literature.

Table I shows the FPGA resources utilized for RDM-QIM implementation after *Place and Route* procedure using a Virtex-4 xc4vsx35 FPGA. Due to the 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.

4 Conclusions

This paper presents an efficient RDM-QIM hardware implementation. The computing time for each stage suggest that in a watermarking-based multi-broadcasting application our implementation will be very adequate. Because RDM-QIM algorithm can be applied to any type of multimedia signals our implementation works very well for audio, images and video signals. Our FPGA implementation is faster than a Digital Signal Processor solution for more than two orders of magnitude.

Acknowledgments

The authors would like to thank CONACyT for financial support.

DOI: 10.1587/elex.6.1064
Received June 11, 2009
Accepted June 24, 2009
Published July 25, 2009