2008 International Conference on Reconfigurable Computing and FPGAs

A Reversible Data Hiding Algorithm for Radiological Medical Images and its Hardware Implementation

Z. Jezabel Guzmán Zavaleta[1], Claudia Feregrino Uribe[2], René Cumplido[3]
National Institute for Astrophysics, Optics and Electronics, INAOE, MEXICO
{zguzman[1], cferegrino[2], rcumplido[3]}@inaoep.mx

Abstract— **In this paper we introduce a reversible data hiding algorithm for medical images and its hardware implementation. Using the advantages of some proved methods and the characteristics of radiological medical images we obtain a large embedding capacity with minimum distortion of the original image using an easy control for recovering the hidden data and the original image. Moreover, in order to speed up computations, a FPGA-based hardware implementation is explored. Due to its simplicity, the proposed architecture can be used as coprocessor or custom core in reconfigurable hardware/software platforms.**

Key words— **Reversible Data Hiding, Radiological Medical Images, Hardware Implementation, FPGAs.**

I. INTRODUCTION

Data hiding is a form of steganography that embeds data into digital media for the purpose of identification, annotation and copyright [1]. Reversibility in data hiding processes is the characteristic that the method can recover exactly the original host signal (without any distortion) upon extraction of the embedded information. For that reason, reversible (lossless) data hiding is widely used on sensitive imagery such as deep space exploration, military reconnaissance and medical diagnosis. In medical domain the main objectives of data hiding are: information protection with application like integrity control and data hiding for the purpose of inserting meta-data to render the image more usable [2].

Recently, several reversible data hiding methods have been proposed; some overviews can be found in [3, 4]. However, not all the methods are suitable for medical images that contain homogeneous regions, where the insertion of information produces several distortions on the embedding host.

In this paper, we propose a new reversible data hiding technique, suitable for radiological medical images, using the advantage of some proved methods, which can embed a significant amount of data while keeping high visual quality. Also the hardware architecture is presented as a mean to speed up computations in embedded reconfigurable platforms that are now commonly used as an efficient alternative in digital signal processing applications.

II. REVERSIBLE DATA HIDING METHODS

Different methods are available for reversible data hiding. The most recent methods that are suitable for medical images have many advantages but also they present disadvantages; for example: some of them have a low distortion of the image but they have a low embedding capacity, or vice versa; in other cases, they have a good performance in embedded capacity and low distortion but they require to save a large amount of data for the exact recovering. In general, most of the publications do not show results with medical images.

Based on the published results, we have chosen three distinctive reversible data hiding algorithms which are the best performers. We tested those methods with radiological medical images and we show their advantages and disadvantages.

A. Lee Algorithm

Lee *et al* [5] propose a reversible image authentication technique based on watermarking. Their method is quite simple and it produces a small image distortion by utilizing the difference between adjacent pixel values. The algorithm scans the image, obtaining the difference between pairs of pixels, the odd-line and the even-line. Depending on the difference value, it can embed data or not, that is, if the difference is equal to -1 or 1 then it embeds a bit, in this case if the bit to be embedded is '0' then the difference value remains equal, otherwise the difference value changes to -2 or 2; if the difference value is equal to 0 then it does nothing, otherwise, it shifts the differences histogram. In order to generate a watermark they combine the hash of the image with a binary logo image using the bit-wise XOR operation. The recovering of the embedded data is performed in the reverse order as the embedding process.

The advantages of this algorithm are: the easy control for recovering the original image and the embedded data; it does not need to save a location map or any other information; the PSNR (Peak-Signal-to-Noise-Ratio) of the watermarked image is greater than or equal to 51 dB, with adequate capacity for addressing many applications; this algorithm is useful for homogeneous images. However, authors do not suggest how they manage underflow and overflow problems that may exist.

B. Ni Algorithm

Ni *et al* [6] proposed a lossless data embedding technique that utilizes the zero or the minimum points of the histogram of an image and modifies the pixel grayscale values to embed data into the image. The complexity of their algorithm is low, and the PSNR between the original and the marked image is greater than 48 dB.

Ni's method first makes the histogram of the image and then checks where the histogram has a zero value, which is called a *zero point* (*Z*), and then it founds a *peak point* (*P*), the maximum value of the histogram. Equation 1 summarizes the embedding process.

$$I_S(i,j)=\begin{cases} I(i,j)-1 & \text{if } Z<P \ \& \ I(i,j)>Z \ \& \ I(i,j)<P \\ I(i,j)+1 & \text{if } Z>P \ \& \ I(i,j)<Z \ \& \ I(i,j)>P \\ I(i,j)+b-1 & \text{if } Z<P \ \& \ I(i,j)=P \\ I(i,j)+b & \text{if } Z>P \ \& \ I(i,j)=P \\ I(i,j) & \text{otherwise} \end{cases} \qquad (1)$$

Where, *I(i,j)* is the original image of size *MxN* pixels; *Z* is the zero point; *P* is the peak point and *b* is the bit to be embed '1' or '0' and I_S is the stego image.

The detection algorithm needs the *Z* and *P* values used. First, the image is scanned then it finds the *Z* and *P* values and shifts back the histogram. At the same time the algorithm has to extract the embedded bits.

With the aim of obtaining a large embedding capacity Ni's method can use more pairs of *Z* and *P* points for each block. Also Fallahpour *et al* [7] propose to divide the image into n-blocks and run the algorithm over each block, and also using *k*-pairs of *Z* and *P* points.

The main advantages of Ni's method combined with [7] are: the simplicity of the embedding process and the hidden data extraction and the method utilizes an easy control for the exact recovery saving a few control bits. However the main disadvantage is that the method has been exploited and is difficult improves its performance.

C. Thodi Algorithm

Thodi *et al* [8] proposed a reversible watermarking algorithm for digital images. It embeds information into prediction errors of adjacent pixels using a MED Predictor (Medium Edge Predictor), see equation (2). Consider a pixel with value *x* and its context; the pixel to its right, bottom and bottom right, *a*, *b* and *c* respectively, then the value predicted and its prediction error is given by (3).

$$x_p=\begin{cases} \max(a,b) & \text{if } c\le \min(a,b) \\ \min(a,b) & \text{if } c\ge \max(a,b) \\ a+b-c & \text{otherwise} \end{cases} \qquad (2)$$

$$p_e = x - x_p \qquad (3)$$

The value of the prediction error p_e and the pixel intensity *x* determine which locations can be embedded by prediction error expansion. In that manner the method organizes the locations in different sets:

- The set *E* contains all locations that can be used for embedding a bit *i;* that is
$$E=\{x: 0\le x+p_e\le L-2 \ \wedge |p_e|<T\}$$
- The set N_e are all locations than can not be expandable; that is
$$N_e=\{x: |p_e|>T\}$$
- The set U_e is the union of *E* and N_e, that is
$$U_e=\{0<x+p_e<T-1 \ \vee \ L-1-T<x+p_e<L-1\}$$
- And the set *U* contains all locations that can not be change.

Where *T* is the threshold and *L* is the representative pixel intensities. The sets *U* and U_e both need flag bits for being recognized; because of this the algorithm reserves some locations for embedding those extra bits.

Thus the insertion of a bit *i* into the referred pixel *x* is performed by error expansion using (4) obtaining the modified pixel value $\tilde{x}$.

$$\begin{aligned} &\text{if} \quad \tilde{p}_e = 2p_e + i \quad \text{then} \\ &\tilde{x} = x_p + \tilde{p}_e = x + p_e + i \end{aligned} \qquad (4)$$

The embedding algorithm proceeds as follows:

1. Preprocess the image to separate the reserve region *R* from the region *S*, that is, the rest of the image; region *R* is used for the extra bits and region *S* for embedding the bit stream. Save the LSB of every pixel on region *R* (R_{LSB})
2. Form the bitstream *B* by combining a payload *Py*, an end-of-payload indicator, *EOP,* and R_{LSB} as $B = Py \cup EOP \cup R_{LSB}$
3. For every pixel in region *S* (without the last row and column), obtain the p_e, and then select the set that it belongs to, *E*, N_e or $U = S - E - N_e$
   - If it belongs to *E* then drop a bit of the beginning of *B* and embed it into *x*. If also it belongs to U_e append a flag bit '1' at the beginning of the bitstream *B*
   - If it belongs to N_e then shift right the *x* value by *T* positions when $p_e \ge 0$ or shift left the *x* value by *T*-1 when $p_e<0$; in that way the decoder will can distinguish the embedded locations. If also it belongs to U_e append a flag bit '1' at the beginning of the bitstream *B*
   - Else (it belongs to *U*) append a flag bit '0' at the beginning of the bitstream *B*

If some bits remain in *B* then embed those bits in the LSB's of *R*.

At the decoder side the watermarked image has to be scanned in reverse order than the embedding process.

Because the decoder extracts the bit embedded and restores the image, it is important to restore the original pixel intensity value at the current location before proceeding to the next location.

The advantage of this algorithm is that it has a great embedding capacity with minimum distortion of the image. However, the disadvantages are the necessity of storing control flag bits, and the preprocessing performed when reserving a region for extra bits, that increases the execution time of the algorithm for large images.

Other quite similar methods are Tian's algorithm [9] and Kallel's improved of Tian's method [10] that uses difference expansion for embedding. The main disadvantage of those methods is the large location map that has to be saved for the lossless recovering. Thodi *et al* [11], made a comparison between his own method versus Tian's method obtaining similar results. The advantage of Thodi's method over their own improved Tian's method is the use of flag bits to recover the image versus a large location map.

III. Proposed method

We propose to use the main advantages of the examined methods and the characteristics of radiological medical images, with the aim of obtaining a large embedding capacity with minimum distortion of the image. Also we use an easy control for the exact recovering of the original image.

Thodi algorithm scans the image and selects the best locations for the embedding process, generates less distortion using prediction error expansion; the challenge is to do it without ambiguity at the recovering process and without saving control bits, i.e. when the method separate the locations in sets, the intersection of these sets is non empty, for that reason the method inserts control bits in order to select the set that the location belongs to. In medical images the pixel intensity values are commonly homogeneous; with that premise, we select the location with the minor difference between adjacent pixels as Lee algorithm. In that manner we embed data only at locations with prediction error values equal to 1 or -1. In the hiding process the pixel values are changed, with the bit insertion or with the shifts, this may cause underflow or overflow in pixels with values on the limits of the gray scale. To avoid underflow or overflow problems, we suggest the use of a pre-processing step based on Ni histogram shifting.

In the pre-processing step the image histogram is obtained and the zero points on the histogram boundaries (Z_f and Z_L) are found then a zero map is made. This process is explained in the next subsections. Once we eliminate underflow and overflow problems, we embed the bitstream B into the image.

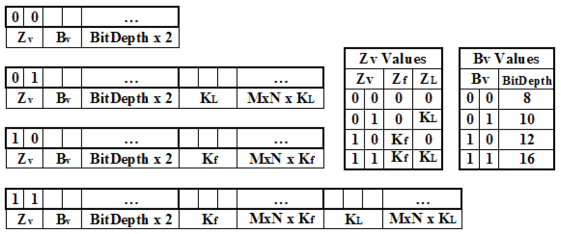
A. Underflow and Overflow Problem

The proposed method utilizes prediction error expansion to embed data, and for managing any problem of underflow/overflow we stretch the histogram of the image. The stretching is made by removing any pixel value that, after some modification, can cause underflow or overflow as follows:

1. For a grayscale image I of M x N pixels, obtain the histogram
2. Find the first and the last zero points (Z_f and Z_L) on the histogram (see Ni method). Form the *Zero Map* (see next subsection).
3. Make the stretching. With every pixel of the image,
   - If $I(i,j) < Z_f$ then shift to the right that is, $I(i,j)+1$
   - If $I(i,j) > Z_L$ then shift to the left that is, $I(i,j)-1$
   - Otherwise do nothing
4. Form the bitstream appending the Zero Map (ZM), the payload (Py) to be embedded and an end of payload flag (EOP): $B = ZM \cup Py \cup EOP$.

B. The Zero Map

The *zero map*, *ZM*, is needed for the lossless recovery process. We append the zero-map at the beginning of the bitstream to be hidden. In case the histogram does not have zero values then we take the minor value as a zero point at each boundary. The first two bits of the map are used to identify zero points, the next two represent the bit depth of the image and the next bits are used for the zero point values (Z_f and Z_L). When the zero point is the minor value different from zero then we save the values of the zero points (K_f / K_L) and its location inside the image (K_f/K_L x M x N) guarantying the lossless recovering. The *ZM* is showed in figure 1.


0 0 …
Zv Bv BitDepth x 2
0 1 … …
Zv Bv BitDepth x 2 KL MxN x KL
1 0 … …
Zv Bv BitDepth x 2 Kf MxN x Kf
1 1 … … …
Zv Bv BitDepth x 2 Kf MxN x Kf KL MxN x KL

Zv Values			
Zv		Zf	ZL
0	0	0	0
0	1	0	KL
1	0	Kf	0
1	1	Kf	KL

Bv Values		
Bv		BitDepth
0	0	8
0	1	10
1	0	12
1	1	16

Figure 1: The Zero Map

C. Embedding Process

This is the process to embed a bitstream B into an Image I, obtaining the modified image I_S with the same dimensions as the original. The last row and the last column of the image must not be used for embedding purposes.

1. Eliminate the underflow/overflow problem in the image (explained before)
2. In a determined order (left to right and up to bottom), scan the image $I(i,j)$ where i= 0 to M-1 and j=0 to N-2 and with every pixel do:
   - If i is equal to M-1 or if j is equal to N-1 then $I_S(i,j) = I(i,j)$
   - Else, obtain the prediction error value p_e of $I(i,j)$ and its context showed in (3),

- If p_e= 1 then embed a bit b from the bitstream B, that is $I_S(i,j) = I(i,j) + b$
- If p_e= -1 then embed a bit b from the bitstream B, that is $I_S(i,j) = I(i,j) - b$

- If $p_e \geq 2$ then make a right shift, that is $I_S(i,j) = I(i,j) + 1$
- If $p_e \leq -2$ then make a left shift, that is $I_S(i,j) = I(i,j) - 1$
- Otherwise $I_S(i,j) = I(i,j)$

D. Extraction and recovery process

The exact recovery process is performed in inverse order than the embedding process, i.e. if the embedding process was in left to right and up to bottom order then the inverse process is going to be bottom to up and right to left order. It obtains the original image $I(i,j)$ from $I_S(i,j)$ following the next steps*:*

1. Scan the image in reverse order and with every pixel of $I_S(i,j)$ (where $i = M$-2 to 0 and $j=N$-2 to 0) do:
   a. Obtain the prediction error value p_e of $I_S(i,j)$ and its context showed in (3),
      - If $p_e = 1$ or $p_e = -1$ then the embedded bit b was '0', append a '0' at the beginning of the recovered bitstream B_R and $I(i,j) = I_S(i,j)$
      - If $p_e = -2$ then the embedded bit b was '1', append a '1' at the beginning of the recovered bitstream B_R and $I(i,j) = I_S(i,j)+1$
      - If $p_e = 2$ then embedded bit b was '1', append a '1' at the beginning of the recovered bitstream B_R and $I(i,j) = I_S(i,j)$-1
      - If $p_e > 2$ then make a left shift, that is $I(i,j) = I_S(i,j) - 1$
      - If $p_e < -2$ then make a right shift, that is $I(i,j) = I_S(i,j) + 1$
      - Otherwise $I(i,j) = I_S(i,j)$

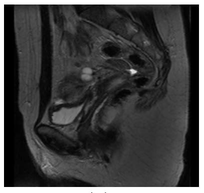
After recovering the embedded bitstream, decode the zero map from the beginning of B_R, see figure 1. Once the Z_f and Z_L values are obtained, if it is necessary the location map, stretch back the histogram of the image I to recover the original image.

E. Other Improvements

We can use the specific characteristics of the radiological medical images to obtain higher embedding capacity. The characteristic are: pixel intensities regularly homogeneous; the image perimeter commonly black, the region of interest for the physician is the center of the image and any alteration of the image perimeter would not alter the medical diagnostic. Thus we propose to use the slight modification of using black 2x2 regions of the image to hide data; i.e. if the context of a referred pixel is a black region and if the referred pixel is also black then we embed a bit on that pixel, else we shift its value to the right. This simple change may increase significantly the embedding capacity on this kind of images. If higher hiding capacity is required, we can use a second embedding process to hide data into the estego image.

IV. Algorithm Evaluation

Even though the published hiding methods show interesting results, not all of them provide results on medical images, so we analyze their performance on some radiological images and we compare the results against our method. We use the PSNR and the hiding capacity metrics. The PSNR reflects the average error produced on the stego image. We use some medical gray scale images (see figure 2) in DICOM format [12] and we hide the standard text *Alice,* from the Canterbury corpus [13], as a bitstream into each image, with a total of 149 KB.


(a)

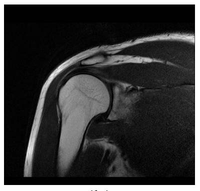

(b)

Figure 2 (a) Image1: MR-MONO2-12-an2.dcm 256x256x12; (b) Image 2: MR-MONO2-12-shoulder.dcm 1024x1024x12

In the tables 1 and 2 we show the obtained results for different images, the labels represent the different methods tested:

Lee: represents the results for the Lee method
Ni: using the Ni improved method with 4 blocks
Thodi: using the Thodi method with threshold parameter of 3
Lee+Sh: are the results with the Lee algorithm plus shift histogram eliminating underflow and overflow problems
P1: the proposed algorithm
P2: P1 and hiding data in black regions
P3: P2 with 2 embedding layers, i.e. we insert the data into the original image, and then we insert data again into the embedded image.

The methods were programmed with Matlab 7.4 R2007a to obtain the comparisons between them. In the tables, we show the hiding capacity measured in bits and the control bits required for every method, the difference between them gives the real hiding capacity. The rows of underflow and overflow show the number of pixels with these problems. Also we measure the execution time in every method measuring only the hiding execution process. The PSNR is given by (5); values over 36 dB are acceptable in terms of degradation, which means no significant degradation is observed by the human eye [14].

$$PSNR = 10 \times \log_{10}\left(\frac{(MaxValue)^2}{MSE}\right) \qquad (5)$$

$$MaxValue = (Bits - 1)^2$$

$$MSE = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}(I_O(i,j) - I_E(i,j))^2$$

Where *MaxValue* is the maximum representative value; MSE is the Mean Squared Error, I_O is the original image and I_E is the estego image.

As it can be seen in tables 1 and 2, the Thodi method shows a good performance in terms of hiding capacity without visible distortions on the medical image. One disadvantage is the increased execution time on big images because the processing time required to find the hiding locations. Other disadvantage is the great number of bits that it needs to save inside the image for the exact recovery, control bits, diminishing the real hiding capacity.

Lee method has an easy way to hide data and recover the original image, the disadvantage is the poor hiding capacity compared with the other methods, as we can see in the tables. The P1 algorithm requires a few control bits for the exact recovery; however it eliminates the underflow and overflow problems that Lee method shows, with a slight more distortion.

With the P2 algorithm we obtained better results than the Lee method but with the P3 we can reach and exceed the other methods results. With P4 we show that we can increase even more the real hiding capacity without visible distortions on the image, also without underflow or overflow problems and recovering the same original image.

TABLE 1: IMAGE1 RESULTS

	Lee	Ni	Thodi	Lee+Sh	P1	P2	P3
Hiding Cap (bits)	467	8308	10309	467	1112	8762	13572
Real Capacity (bits)	467	7918	2467	439	1084	8734	13486
PSNR (dB)	75.91	81.68	65.48	74.85	72.36	72.23	65.99
ExecTime (seg)	0.01	0.08	22.09	0.02	0.07	0.07	0.15
Overflow (pixels)	0	0	0	0	0	0	0
Underflow (pixels)	256	0	0	0	0	0	0
Control (bits)	0	390	7842	28	28	28	86

TABLE 2: IMAGE2 RESULTS

	Lee	Ni	Thodi	Lee+Sh	P1	P2	P3
Hiding Cap (bits)	22357	226762	270013	22357	62707	278482	453165
Real Capacity (bits)	22357	226564	19367	22329	62679	278454	453079
PSNR (dB)	76.57	72.79	67.21	70.87	69.95	69.76	63.56
ExecTime (seg)	0.10	0.53	15222.36	0.47	1.21	1.32	2.70
Overflow (pixels)	0	0	0	0	0	0	0
Underflow (pixels)	2733	0	0	0	0	0	0
Control (bits)	0	198	250646	28	28	28	86

V. HARDWARE IMPLEMENTATION

In this section, the architecture of the proposed method is elaborated. We first provide high level description of the encoder, followed by its architectural details.

Figure 2 presents the block diagram of the architecture. The AGU block generates the access addresses to the memory to obtain the pixel values for the hiding process block. The AGU block also generates the next address for the payload memory, obtaining the hiding bit for every location depending on the *embed_flag*. The *embed_flag* indicates if there was a hidden bit or not.

When the entire locality is obtained, i.e. the referred pixel x and its context a, b, c; the next block processes those pixels and it obtains the modified pixel value, and also the embedded flag.

A. Hiding Process Block

The input data to the Hiding process are the referred pixel, its context and the bit to be hidden, in case that that bit is required. The Check_Context0 block verifies if the referred pixel and its context is a black region or not, then it generates a *hiding_flag* that indicates every possible situation. The prediction error block, evaluates the input values and it obtains the prediction error value.

The control signals block only summarizes the comparators used for the selection of the operation to carry out, that is, if the pixel value is going to be modified with the bit to be hidden or if the pixel value requires a left or right shifting. The output values are the modified pixel value *x_new* and the *embed_flag*.

The extraction and recovery design is similar to the hiding process, obtaining the original value for the pixel and the hidden bit.

The hardware design was modeled using VHDL and the synthesis using Xilinx ISE 9.2i tools targeting Xilinx Spartan 3E technology with xc3s500e-5-fg320 target device. Table 3 presents the summary of the synthesis results.

TABLE 3: SUMMARY OF THE SYNTHESIS REPORT OF THE HIDING PROCESS UNIT

Clk Period (ns)	18.913
Cells Usage (BELS)	467
Slices	151
Slice FFs	81
LUTs	239

The implementation test was made with Simulink/ Matlab using the hardware-in-the-loop technique. The test step consists in processing the same images into the software and hardware implementations. We compare the two obtained results for each image and in all cases the modified image was the same.

With the aim of comparing the hiding process execution time in both software and hardware implementations; we programmed the proposed method in C language and VHDL.

The execution time results from the images presented before show that the time performance is better in the hardware implementation approximately 10 times. For example, for image 1 the execution time for the hardware and software implementations is 0.00123 and 0.017317 respectively. These results can be easily extrapolated to other images as the execution time only depends of the image size.

VI. CONCLUSION

We proposed a new reversible method based on other proved techniques. We reached and exceed the performance of the others tested methods, giving an easy way to hide a large payload without visible distortions, with high levels of PSNR values, and with a simple solution to the underflow and overflow problem.

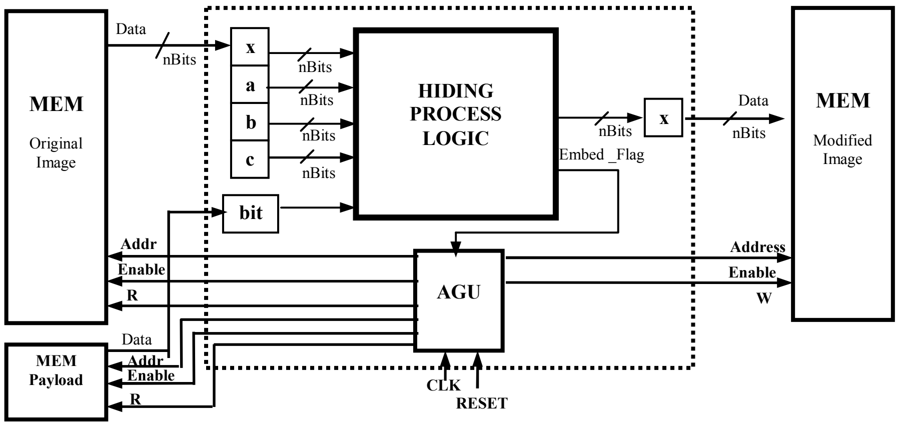


Figure 3 Block Diagram of the Proposed Method

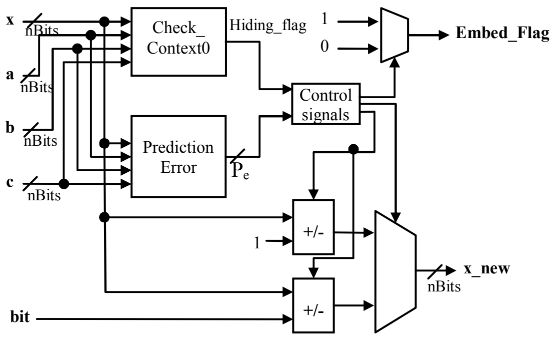


Figure 4 Block Diagram of the Hiding Process

As a preliminary result, a hardware implementation has been done, showing the plausibility of using the algorithm in embedded medical systems. The hardware implementation presented improves the performance of the proposed method compared with the software implementation. The hardware implementation is over 10 times faster than the software programmed in C language. These results are obtained from the basic design of the architecture results using the test images.

The hardware implementation is suitable to be used as a coprocessor due to its rather simple control and data exchange mechanism and its ability to provide parallel processing.

In one field of growth, more work has to be done in order to define an efficient hardware architecture to fulfill additional constrains such power consumption and mobility.

ACKNOWLEDGMENTS

The first author thanks the support given by the CONACyT, *Consejo Nacional de Ciencia y Tecnología*, for the support given by the scholarship No. 216817.

REFERENCES

[1] Information Hiding: First International Workshop, R. J. Anderson, Editor, Lecture Notes in Computer Science 1174, Isaac Newton Institute, Cambridge, England, Springer-Verlag May 1996.
[2] G. Coatrieux, L. Lecornu, B. Sankur, C. Roux, A review of Image watermarking applications in healthcare, EMBC06, New York, USA, Sept. 2006.
[3] Awranjeb M. An Overview of Reversible Data Hiding. International Conference on Computer and Information Technology (ICCIT) Dec 19-21,2003. Jahangirnagar University. Bangladesh, pp 75-79.
[4] Y. Hu, B. Jeon, Z. Lin, H.Yang. Analysis and Comparison of Typical Reversible Watermarking Methods. Y.Q. Shi and B. Jeon (Eds.): IWDW 2006, LNCS 4283, 2006.c Springer-Verlag Berlin Heidelberg, pp. 333–347.
[5] Lee S-K, Suh Y-H, Ho Y-S. Reversible Image Authentication Based on Watermarking. IEEE International Conference on Multimedia & Expo (ICME) 2006, Canada, pp 1321-1324.
[6] Ni Z., Shi Y., Ansari N., and Su W., "Reversible data hiding," *Proc. ISCAS 2003*, vol. 2, pp. 912–915.
[7] Fallahpour M, Sedaaghi M. "High Capacity lossless data hiding based on histogram modification". IEICE Electronic Express, Vol. 4, No. 7, April 10, 2007, pp. 205-210.
[8] Thodi D.M. and Rodriguez J. J., "Prediction-error-based reversible watermarking," in *Proc. IEEE Conf. Image Processing*, Oct. 2004, pp. 1549–1552.
[9] Tian, J.: Reversible data embedding using a difference expansion. IEEE Trans. On Circuits and Systems for Video Technology. Vol. 13, No. 8, Aug. 2003, pp. 890-896.
[10] Kallel I., Bouhlel M.S, Lapayre J.C. Improved Tian's Method for Medical Image Reversible Watermarking GVIP Journal, Volume 7, Issue2, August 2007
[11] Thodi DM, Rodriguez J. Expansion Embedding Techniques for Reversible Watermarking. IEEE Transactions on Image Processing. Vol 16, No 3, March 2007
[12] S. Barré: Medical Image Samples: *http://barre.nom.fr/medical/samples/*
and MATLAB Central File Exchange: *http://www.mathworks.ch/matlabcentral/fileexchange/loadFile.do?objectId=2762&objectType=FILE*
[13] The Canterbury Corpus: http://corpus.canterbury.ac.nz/
[14] Xuanwen Luo, Qiang Cheng, Joseph Tan, "A Lossless Data Embedding Scheme For Medical in Application of e- Diagnosis," *Proceedings of the 25th Annual International Conference of the IEEE EMBS Cancun*, Mexico. September 17-21, 2003