“Algoritmo Reversible para Ocultamiento de Datos y su Arquitectura Hardware”

por

Zobeida Jezabel Guzmán Zavaleta

Tesis sometida como requisito parcial para obtener el grado de

MAESTRA EN CIENCIAS COMPUTACIONALES

en el

Instituto Nacional de Astrofísica, Óptica y Electrónica

15 de Octubre de 2008

Tonantzintla, Puebla.

Supervisada por

Dra. Claudia Feregrino Uribe

Investigadora titular del INAOE

Resumen

En esta tesis se propone un método para ocultamiento de datos sobre imágenes sin pérdida (reversible); así como el diseño de su arquitectura hardware. La característica reversible, se refiere a la capacidad del método de recuperar la imagen original después de haber recuperado la información oculta. Este algoritmo se basa en las ventajas de algunos métodos probados, obteniendo gran capacidad de ocultamiento y con buena calidad de la imagen.

Siendo reversible puede ser utilizado para ocultar datos en imagenología del tipo sensible, tal es el caso de imágenes de exploración espacial profunda, de reconocimiento militar y de diagnóstico médico. Los métodos de ocultamiento de datos con pérdida generalmente no son utilizados en este tipo de imágenes, ya que los usuarios se preocupan por no perder la información original.

Logros de esta tesis: 1) se exploraron las ventajas y desventajas de los algoritmos tomados como base sobre imágenes radiológicas, 2) algoritmo propuesto para ocultamiento de datos y algunas modificaciones para mejorar el rendimiento en imágenes radiológicas, 3) arquitectura hardware del algoritmo propuesto.

Abstract

This thesis explores a reversible data hiding method for images and its hardware architecture design. Reversible data hiding is a particular kind of data hiding. Reversibility is the characteristic that the method can recover exactly the original host signal (without any distortion) upon extraction of the embedded information. Using the advantage of some proved methods, the proposed algorithm can embed a significant amount of data while keeping high visual quality.

Reversible (lossless) data hiding is widely used on sensitive imagery such as deep space exploration, military reconnaissance and medical diagnosis. In applications with this kind of images is necessary to preserve the original data integrity, for that reason other types of data hiding are not suitable.

The contributions accomplished in this project are: 1) the advantage and disadvantage exploration of existing methods for radiological medical images, 2) a new reversible data hiding algorithm including some improvement for achieving a better performance for radiological medical images, 3) a hardware architecture for the proposed method.

Agradecimientos

Primeramente agradezco a mi asesora la Dra. Claudia Feregrino Uribe por haberme guiado en todo el camino hasta terminar este proyecto, por brindarme de su valioso tiempo y compartir su experiencia, por corregirme cuando fue necesario, por su ejemplo y amistad, gracias.

También un especial agradecimiento a mis revisores que siempre estuvieron dispuestos a guiarme, el Dr. Jesús Ariel Carrasco Ochoa, Dr. René Armando Cumplido Parra y el Dr. Carlos Alberto Reyes García.

A mis compañeros de clase, a todas las personas que me ayudaron directa o indirectamente incluyendo a todo el personal administrativo del instituto a lo largo de la maestría, gracias.

Gracias a la ayuda brindada por el CONACYT con la beca otorgada No. 216817, sin duda una gran alivio.

Al INAOE por abrirme sus puertas y permitirme estudiar en esta excelente institución, gracias.

Me siento muy agradecida con mi familia que tanto me apoyó y brindó su ayuda incondicional; gracias a mi esposo Jared, a mi paciente hija Abish, a mi madre Zobeida, a cada uno de mis hermanos Sariah, Victor, Gibran y Ari, a mi suegros Carmen y Jaime, a mis cuñadas Carmen y Paty.

Gracias Padre Celestial por ser tu hija, por todo tu amor y por poner en mi camino a todas las personas que me ayudan a crecer.

Gracias.

Para Abish,

mi amorosa y paciente hija.

Tabla de Contenido

CAPÍTULO 1
INTRODUCCIÓN

1.1. Problemática Actual

Con los avances en la tecnología hoy en día es muy sencillo transferir información de un lugar a otro mediante medios electrónicos. Nuevas tecnologías y nuevas aplicaciones traen consigo nuevas amenazas por lo que son necesarios nuevos sistemas de seguridad. La criptografía se volvió importante en las transferencias de información mediante las redes de computadoras. Sin embargo, las tecnologías más recientes en la búsqueda de la seguridad se basan en ocultamiento de la información.

La aplicación clásica de las técnicas de ocultamiento de información es la comunicación secreta. El ocultamiento de datos también es usado en el contexto de privacidad; brindando anonimato a proveedores de Internet, como mecanismo de comunicación entre dispositivos, para ocultar meta-datos arbitrarios en imágenes, para ocultar canales alternativos en transmisiones de televisión, etc. [1].

Una importante subdisciplina del ocultamiento de la información es la esteganografía. Mientras la criptografía protege el contenido del mensaje volviéndolo inteligible, la esteganografía oculta el mensaje de tal manera que

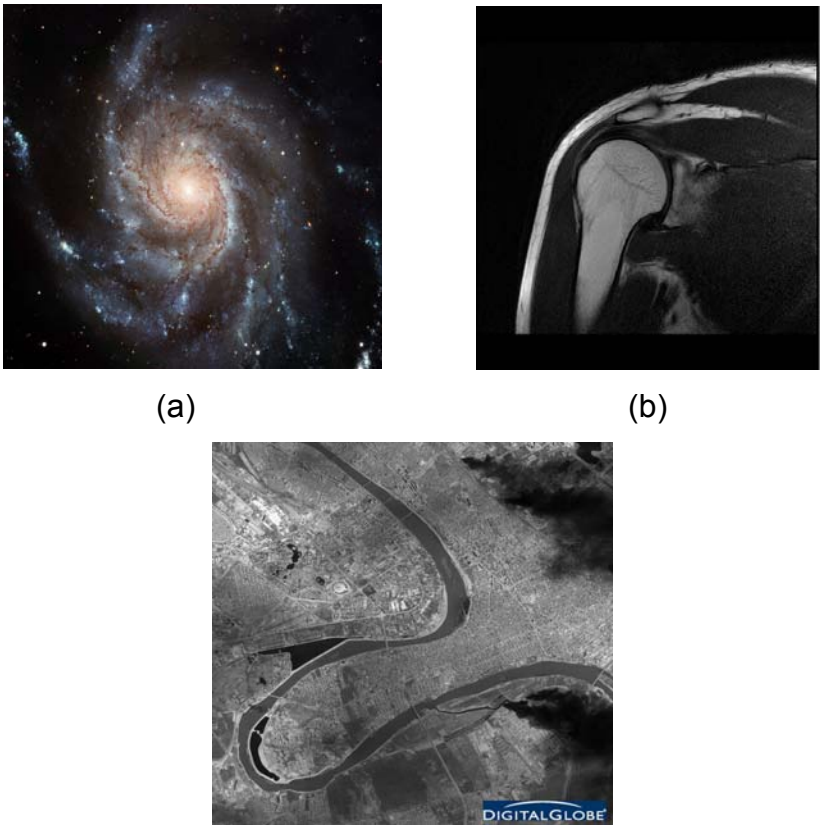
pase inadvertido. Así la esteganografía puede ser usada para transmitir mensajes ocultos.

Una forma de esteganografía ampliamente adoptada son las marcas de agua. La información que se oculta mediante un sistema de marcas de agua siempre está asociada al medio en el que se está ocultando mientras que los métodos esteganográficos simplemente ocultan información. Las marcas de agua poseen requisitos dependientes de la aplicación que son: imperceptibilidad en caso de ser marcas de agua no visibles, robustez o fragilidad y capacidad de ocultamiento. La imperceptibilidad se refiere a que la marca de agua debe pasar desapercibida para un observador humano, es decir, que la distorsión que se genera al insertar los datos sea mínima. La capacidad se define como la cantidad de información que se puede ocultar como marca de agua en algún medio de forma fiable. La robustez de una marca se refiere a la dificultad de degradarla y hacerla recuperable, es decir ante un ataque (como rotación de la imagen o cambio de contraste de la imagen, etc.) la información oculta no debe perderse. Así el marcado puede ser frágil: si lo que interesa es el control de integridad de las imágenes, y si se pierde la información marcada sea posible identificar si la imagen fue alterada; o robusto: para propósitos de autenticación.

Con las técnicas de seguridad mencionadas es posible ocultar información en medios digitales multimedia, como audio, video, texto e imágenes. En el caso de las imágenes existe el tipo de imagenología sensible. La imagenología sensible es aquella en la que es muy importante mantener los datos originales sin distorsiones, tal es el caso de las imágenes de exploración espacial, de reconocimiento militar y de diagnóstico médico. La figura 1 muestra algunos ejemplos de imagenología sensible.

Las imágenes de diagnóstico médico se utilizan en gran número hoy en día debido a las aplicaciones actuales de teleradiología, telemedicina y repositorios de imágenes médicas. La teleradiología se refiere al envío de imágenes radiológicas a distancia entre diferentes hospitales para su

diagnóstico. La telemedicina requiere de envíos de datos clínicos de un centro a otro, facilitando la toma de decisiones por parte del especialista.



(a) (b)

(c)

Figura 1. Imagenología sensible.
(a) Exploración espacial, (b) Imagen médica radiológica, (c) Reconocimiento militar [2]

Los repositorios de imágenes médicas se encuentran disponibles en la Web, para realizar pruebas a algoritmos y herramientas de ayuda al diagnóstico.

Estas aplicaciones requieren de sistemas de seguridad encargados de mantener confidencial la información de los pacientes, así como la integridad de los datos transmitidos, etc. Las técnicas de seguridad mencionadas se han utilizado con estos propósitos, pero debido a la naturaleza sensible de

estas imágenes es necesario poner gran interés en la imperceptibilidad de los datos ocultos y se requiere además mantener íntegra la imagen original después de haber extraído los datos ocultos. Por ello un tipo de ocultamiento, ya sea método esteganográfico o de marcas de agua ampliamente usado en la imagenología sensible es el tipo reversible.

El ocultamiento de datos reversible o sin pérdida es un tipo particular de ocultamiento. La característica de reversibilidad permite que una vez que se hayan extraído los datos ocultos del medio utilizado, sea posible recuperar el medio original, sin ninguna distorsión.

Se ha propuesto una gran cantidad de métodos de ocultamiento reversible [15-27], incluso se han realizado trabajos analizando el rendimiento de los métodos propuestos [15-17]. Sin embargo no todos los métodos son apropiados para ser utilizados en todos los tipos de imagenología sensible; por ejemplo, las imágenes médicas contienen regiones con intensidad de píxeles homogéneas y al ocultar los datos con algunos métodos se producen distorsiones visibles, tales como ruido de sal y pimienta [17]. Por otro lado los métodos apropiados para estas imágenes varían en cuanto a su rendimiento, por ejemplo, algunos generan gran capacidad de ocultamiento pero generan mayor distorsión en la imagen marcada o viceversa [16], existen otros que proporcionan gran capacidad de ocultamiento y sin distorsiones visibles pero su procesamiento es exhaustivo volviéndolos lentos al realizar el procesamiento sobre imágenes de tamaño grande, por ejemplo de 1024 x 1024 píxeles [21, 23].

Las herramientas de software disponibles comercialmente para ocultamiento de datos tienen muchas restricciones en especial para la imagenología de tipo sensible, las cuales presentan formatos especiales, tamaños diversos y con distintas representaciones de píxeles, donde cada píxel es representado por 8, 10, 12, 16 ó 24 bits.

1.2. Motivación

Debido al gran número de métodos propuestos para ocultamiento de datos reversibles es necesario un estudio para elegir un método que tenga las características requeridas para utilizarlo sobre imagenología sensible, tales como imágenes médicas. Las características requeridas dependen de la aplicación específica, pero en general se buscan las siguientes: Reversibilidad para garantizar que la información original se pueda recuperar. Gran capacidad de ocultamiento, algunas aplicaciones insertan meta-datos en las imágenes, lo que requiere considerable capacidad de ocultamiento; por otro lado, algunas aplicaciones insertan cierta información, como logotipos, en toda la imagen para control de integridad, así que entre más información se pueda ocultar es mejor. Menor distorsión sobre la imagen original, para que pase desapercibido el hecho de que la imagen tiene información oculta, es decir que no genere sospechas. También que sea adecuado para imágenes con valores de píxeles homogéneos, esto debido a que existe gran variedad de imágenes, en especial médicas, que tienen esta característica.

Algunos autores han realizado estudios semejantes pero no incluyen resultados al utilizar los diversos métodos en imagenología sensible. Es necesario realizar un estudio que muestre resultados en cuanto a distorsión de la imagen, capacidad de ocultamiento, etc. sobre este tipo de imágenes. Con este análisis se podrían identificar las ventajas y desventajas de cada método para lograr así una sinergia entre los mejores algoritmos analizados generando uno con mejor rendimiento.

Se requiere de un método capaz de ocultar datos en imagenología sensible de manera rápida, aún en imágenes de gran tamaño (1-5 MB). Otro requerimiento es que el método oculte una gran capacidad de bits sin distorsionar la imagen de manera visible y que tenga la capacidad de ser reversible. Aún cuando muchos métodos ya propuestos se centran en

características específicas, es necesaria una implementación que cuente con todas las características ya mencionadas.

Los algoritmos de ocultamiento de información se pueden diseñar en software o hardware. Sin embargo, el hardware ofrece ventajas sobre el primero en términos de área, bajo tiempo de ejecución y a un bajo consumo de potencia [3]. Además se tiene la ventaja que el diseño de hardware se puede empotrar en los dispositivos digitales de tal manera que la imagen marcada se obtenga directamente del origen.

Existen varias alternativas para implementar un diseño en hardware entre las que se encuentran los circuitos integrados de aplicación específica ASICs, microprocesadores, procesadores digitales de señales DSPs y los arreglos programables de compuertas FPGAs. Aunque los tres primeros son muy rápidos y poderosos tienen un costo de fabricación elevado que se justifica cuando la producción de circuitos es muy grande o cuando los niveles necesarios de desempeño son tan altos que no pueden ser alcanzados con la tecnología existente. Por su parte los FPGAs (*Field Programable Gate Array*), permiten diseñar implementaciones que procesen información en tiempo real requiriendo un bajo costo de inversión, con la capacidad de realizar muchas pruebas y con gran velocidad de procesamiento, lo que los hace una alternativa eficiente para diseñar y probar una nueva implementación antes de su posible fabricación en serie.

1.3. Objetivo General

Proponer un método de ocultamiento de datos reversible sobre imágenes sensibles y demostrar su desempeño mediante su diseño e implementación en hardware.

1.3.1. Objetivos Específicos

- Analizar las ventajas y desventajas de los métodos actuales para ocultamiento de datos reversible.
- Proponer un método que cubra con las necesidades de la imagenología del tipo sensible.
- Implementar el método propuesto en software y hardware mostrando los resultados obtenidos.

1.4. Organización de la Tesis

El presente documento se encuentra dividido en 6 capítulos en los cuales se abordarán temas específicos iniciando con una introducción al presente trabajo. En el capítulo 2 se revisa el marco teórico en cuanto a las técnicas de seguridad disponibles dando énfasis al ocultamiento de datos. Se abordan las subdisciplinas del ocultamiento de la información. También se analizan las características del ocultamiento de información del tipo reversible, así como sus aplicaciones.

En el capítulo 3, se estudian algunos métodos ya publicados. Se muestran sus principales ventajas y desventajas. Se exponen algunos resultados sobre imágenes radiológicas que son un tipo de imagenología sensible al programar los métodos en software.

El capítulo 4 presenta el método propuesto para ocultamiento de datos sin pérdida. De la misma manera se exponen algunas mejoras al método propuesto específicas para las imágenes radiológicas.

Las implementaciones software y hardware del método propuesto se verán en el capítulo 5. Se presentarán los resultados obtenidos y por último se dan las conclusiones en el capítulo 6.

CAPÍTULO 2

REVISIÓN DEL MARCO TEÓRICO

El aumento de los sistemas de comunicación y las computadoras originó la creciente demanda de servicios de seguridad. En la actualidad existen diversas técnicas de seguridad que se derivan principalmente de la esteganografía y la criptografía. Generalmente estas disciplinas suelen confundirse ya que ambas forman parte de los sistemas de seguridad, pero son distintas tanto en su forma de implementar como en su objetivo mismo.

La esteganografía es el arte y la ciencia de la información oculta. Un sistema esteganográfico oculta información en un portador de manera que no sea advertido el hecho mismo de su existencia y envío. Por otro lado, la criptografía es utilizada para cifrar o codificar información de manera inteligible para un probable intruso, a pesar del conocimiento de su existencia.

Aún cuando las técnicas de esteganografía y criptografía son distintas, para dar un nivel de seguridad extra a la información ambas técnicas pueden complementarse. El mensaje a ocultar con métodos esteganográficos puede ser previamente cifrado; de tal modo que a un eventual intruso no sólo le costará advertir la presencia misma de la información oculta, sino que si la llegara a obtener, la encontraría cifrada.

A continuación se analizan algunas características de las técnicas de criptografía y esteganografía, enfocándonos principalmente en los métodos de ocultamiento de datos.

2.1 Criptografía

La criptografía provee comunicación secreta sobre medios inseguros. La criptografía es el estudio de técnicas matemáticas relacionadas con los aspectos de seguridad de la información.

La criptografía tiene como objetivo cubrir los siguientes servicios de la seguridad [4]:

- Confidencialidad: este servicio se utiliza para mantener la información transmitida de manera que personas no autorizadas no tengan acceso a ella. Estos servicios van desde la seguridad física de la información hasta algoritmos que manipulan la información de tal manera que sea inteligible.
- Integridad de los datos: se encarga de mantener los datos sin alteraciones por personas no autorizadas. En caso de que exista alguna modificación, ya sea inserción de datos, eliminación o sustitución de la información es necesario detectarla.
- Autenticación: se encarga de identificar a las entidades que transmiten los datos y a los datos mismos.
- No repudio: este servicio previene que una entidad niegue haber realizado algunas acciones o trasferencias.

2.2 Ocultamiento de Datos

El ocultamiento de la información (u ocultamiento de datos) tiene como principal objetivo mantener las comunicaciones secretas y tiene un amplio

rango de aplicación. Por ejemplo, las técnicas de ocultamiento de datos se usan en los sistemas tácticos militares para prevenir que los trasmisores sean localizados.

Una subdisciplina del ocultamiento de la información es la esteganografía. Esteganografía literalmente significa “escritura encubierta” y se deriva de los vocablos griegos *esteganos,* que significa cubierto u oculto y *graphos*, que significa escritura [5].

2.2.1 Breve Historia

Desde la antigüedad se ha utilizado la esteganografía para ocultar información. Algunos ejemplos se toman de los griegos del libro de *Las Historias* en donde Herodoto relata que utilizaba técnicas de ocultamiento de información en tiempos de guerra. Un ejemplo es que tomaban cuadernillos de tablillas de madera, les quitaban la cera que los cubría y grababan en ellas un mensaje, posteriormente volvían a cubrir la tablilla con cera, cuando el mensaje oculto llegaba a su destino éste podía ser leído quitando la cera nuevamente. Con el mismo fin de enviar mensajes ocultos, rasuraban la cabeza de los mensajeros y tatuaban el mensaje oculto en sus cabezas, después que les crecía el cabello los mandaban a su destino. Esta técnica fue utilizada también por espías alemanes al inicio del siglo XX [44].

Hay otros registros en donde los mensajeros llevaban ocultas letras en botones y aretes; textos escritos en maderas pintadas y notas llevadas por palomas. También ocultaban los mensajes en textos marcando micro puntos.

Durante las Guerras Mundiales se adoptó el uso de tintas invisibles, las cuales eran compuestas por sustancias orgánicas y develadas con calor [6].

Muchísimos otros ejemplos de esteganografía pueden encontrarse a lo largo de la historia. Entre los siglos XVI y XVII se publicaron muchos libros de esteganografía, los cuales se basaban principalmente en los primeros libros

sobre criptografía y esteganografía: *Polygraphiæ* y *Steganographia* propuestos por Trithemius, la portada de este libro se presenta en la figura 2.

Figura 2. Portada del libro *Steganographia* de Trithemius [42]

Estos libros los toma Gaspar Schott y los expande presentando el libro *Schola Steganographica* en el año 1665 (véase figura 3) donde muestra el uso de 40 tablas que contienen 24 entradas, una para cada letra del alfabeto, en 4 idiomas: alemán, latín, italiano y francés.

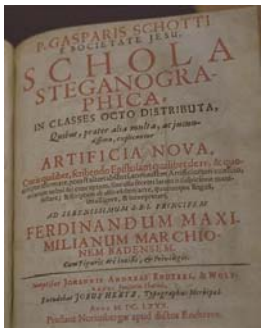


Figura 3. Hoja de título del libro *Schola Steganographica* [43]

Cada letra es correspondida y reemplazada por una palabra o frase, al final el texto obtenido parece una plegaria u oración. También explica como esconder mensajes en música, donde cada nota corresponde a una letra [6]. Otros métodos se basan en la ocurrencia de las notas. La figura 4 muestra este método usado por varios autores.

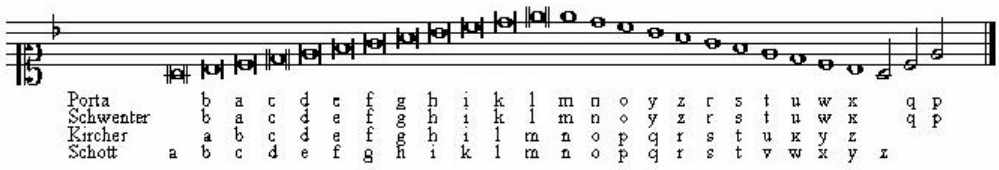


Figura 4. Abecedario con notas musicales según varios autores [6]

Ya en el siglo XX, fueron los inicios del ocultamiento de datos electrónico. Un gran número de compañías han experimentado el uso de señales ocultas para propósitos de control de dispositivos. Desde Muzac Corporation (desde 1954 hasta los 80's) hasta Divx Corporation [7]

El interés en ocultamiento de datos sobre medios digitales que iniciara desde los años 90's sigue en aumento, especialmente existe un crecimiento en el uso de Marcas de Agua para propósitos de protección de derechos de autor, entre muchas otras aplicaciones.

2.2.2 Principios de Ocultamiento de Datos Digital

El modelo de ocultamiento de datos requiere de un archivo inocuo de tipo multimedia en el que se ocultarán los datos, por ejemplo, una imagen; este archivo es el *estego-medio*. También se necesita el mensaje a ser ocultado, este mensaje puede ser de tipo texto, texto cifrado, una imagen o alguno otro que pueda ser representado como una secuencia de bits. Cuando se oculta el mensaje en el *estego-medio* se obtiene la *estego-imagen*, o el *estego-audio*, etc. También se puede utilizar una *estego-llave* (una clave especial) para realizar el proceso de ocultamiento y posteriormente la extracción de los datos.

Las marcas de agua son otra forma de esteganografía. Ambas esteganografía y marcas de agua tienen el objetivo de ocultar datos dentro de estego-medios de manera imperceptible. La diferencia es que la esteganografía sólo se relaciona en las comunicaciones entre dos partes y

generalmente son muy susceptibles a incluso pequeñas modificaciones en el medio que lleva los datos ocultos, perdiendo así parte o toda la información confidencial. Por otro lado, las marcas de agua tienen ciertas características adicionales para garantizar su supervivencia ante ataques. Esto es, dependiendo de la aplicación, pueden ser muy susceptibles a cambios, o por el contrario, se insertan los datos con redundancia para asegurar que la marca oculta no se pierda. Generalmente se usan marcas de agua para aplicaciones donde se involucran muchos usuarios con acceso a la imagen marcada.

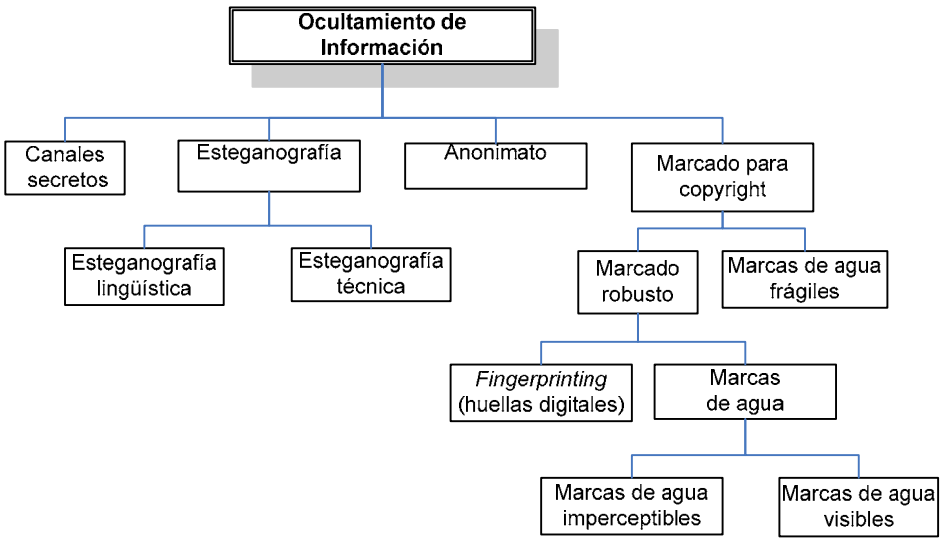


Figura 5: Clasificación de las técnicas de ocultamiento de información [8].

Existen muchas técnicas de ocultamiento de información en imágenes, pero las principales técnicas se basan en los siguientes tres enfoques [6]:

- Inserción en el bit menos significativo: es la manera más sencilla de ocultar información; pero al mismo tiempo la más vulnerable a pérdidas incluso con la más mínima manipulación. Es básicamente tomar el valor del píxel en su forma binaria y reemplazar el bit menos

significativo con el bit del mensaje a ocultar. Esta técnica se puede hacer en distintos canales de la imagen, si se trata de una imagen a color, o también utilizando ciertos criterios de selección de los píxeles en los que se ocultarán los bits.

- Enmascaramiento y filtrado: esta técnica es para ocultar marcas de agua, las cuales incluyen información como derechos de autor, de propiedad o licencias. A diferencia de la esteganografía donde el mensaje oculto es lo importante, en las marcas de agua el objetivo es el estego-medio, añadiéndole a la imagen atributos extra.
- Algoritmos y Transformaciones: esta técnica oculta datos mediante funciones matemáticas, de manera que el mensaje quede insertado en los bits de los píxeles menos importantes. Son ampliamente usadas la transformada Discreta del Coseno (DCT), la transformada de Fourier, y las trasformadas Wavelets. También se manipulan algunas propiedades de las imágenes como la luminancia.

Hasta ahora se han mencionado técnicas esteganográficas y de marcas de agua como subdisciplinas del ocultamiento de datos, pero este documento se enfoca al ocultamiento de datos en general. Así mismo, la aplicación del ocultamiento de datos en este documento está orientada a las imágenes digitales, especialmente la imagenología de tipo sensible, así que de ahora en adelante en lugar de referirnos a los estego-medios en general, hablaremos de técnicas de ocultamiento sobre imágenes.

2.2.3 Aplicaciones de Ocultamiento de Datos

En un gran número de aplicaciones se requiere ocultar datos, por ejemplo, en los sistemas de seguridad de organizaciones militares. En estos sistemas son necesarias las comunicaciones de manera discreta; en donde

más allá de proteger la información con medios criptográficos es necesario que pasen inadvertidos. Se tiene además comunicación anónima.

La industria de la salud y en especial los sistemas de imagenología digital se ven beneficiados por el ocultamiento de datos. Las imágenes médicas se manejan en un formato estándar denominado DICOM (*Digital Imaging and Communications in Medicine*), para manipular, almacenar, imprimir y transmitir información de las imágenes médicas. El formato DICOM separa los datos de la imagen, tiene un encabezado con la información del registro médico del paciente, como es el nombre del paciente, la fecha y el nombre del especialista. Algunas veces se pierde la liga entre la imagen y los datos del paciente; así que una manera de usar una medida de seguridad al respecto sería ocultando los datos del paciente dentro de la imagen. Más aún, se puede utilizar esta técnica para mantener confidencial esa información. Otra técnica emergente relacionada con el sector de la salud es ocultar mensajes en secuencias de ADN. Esto puede ser usado con el objetivo de proteger la propiedad intelectual en la medicina, biología molecular y genética.

En el contexto de aplicaciones multimedia se tienen: monitoreo automático de material con derechos de autor en la Web; evaluación automática de transmisiones de radio y tv; aumento de datos, por ejemplo, anotaciones, información de compras, de otros canales, etc; para prevenir o detectar modificaciones no autorizadas; entre muchas otras [6].

2.3 Ocultamiento de Datos Reversible

El ocultamiento de datos reversible o sin pérdida es un tipo particular de ocultamiento. Este término se emplea desde inicios del año 2000 donde algunos autores propusieron esquemas de este tipo [16].

La característica de reversibilidad indica que aún después de la extracción de la información oculta es posible recuperar la imagen original,

sin distorsión alguna. Por esta razón, los métodos reversibles de ocultamiento se utilizan en imagenología sensible, ya que en este tipo de imágenes es necesario mantener la integridad de los datos.

Recientemente cada vez más y más métodos de ocultamiento reversible se han propuesto, al mismo tiempo, algunos autores han presentado algunas revisiones y análisis de rendimiento a los métodos propuestos, por ejemplo [15-17]. Sin embargo, no todos los métodos son adecuados para todo tipo de imágenes; por ejemplo las imágenes médicas contienen regiones con intensidades de píxeles homogéneas y al insertar los datos a ocultar genera grandes distorsiones en la estego-imagen.

Yongjian Hu [16], menciona en su análisis que uno de los primeros trabajos lo realizó Honsinger [18], y a partir de éste muchos otros métodos se han propuesto. Muchos de los métodos propuestos se basan también en los primeros trabajos de Fidrich y Tian [19-21].

Con base en los primeros métodos de ocultamiento de datos reversibles se han propuesto algoritmos más sofisticados y con mejor rendimiento.

Awranjeb en su estudio [15] clasifica los métodos reversibles dependiendo de la técnica que cada uno utiliza y a la vez menciona las ventajas y desventajas de cada uno. Un resumen de su clasificación y sus observaciones se encuentra a continuación.

- **Los que realizan compresión sin pérdida y cifrado de algunos planos de bits**: las ventajas de este método es que se puede emplear para autenticar imágenes con compresión y la seguridad depende del método criptográfico utilizado para cifrar. Desventajas: un solo plano de bits no ofrece la suficiente capacidad para ocultar un mensaje grande y si se utilizan dos o más planos entonces los datos ocultos se vuelven visibles.
- **Ocultamiento en bajos niveles de píxeles:** es una generalización de inserción en LSB. Ventajas: es un algoritmo simple y es posible incrementar la capacidad de ocultamiento. Desventajas: la capacidad

depende del tipo de imagen, las imágenes suaves brindan mayor capacidad de ocultamiento que las de texturas irregulares y por otro lado si se incrementa la capacidad incrementa también la distorsión de la imagen.

- **Interpretación circular de transformaciones biyectivas:** Ventajas: el método proporciona gran capacidad de ocultamiento con buena calidad de la imagen, sin embargo tiene como desventaja que es un algoritmo complejo y es necesario almacenar información extra para la exacta recuperación de la imagen original, disminuyendo así la capacidad real de ocultamiento.
- **Basados en transformadas wavelet enteras:** Ventajas: gran capacidad de ocultamiento y utiliza una estego-llave lo que incrementa la seguridad. Desventajas: para obtener gran capacidad de ocultamiento se requiere de múltiples planos lo que aumenta la distorsión de la imagen.
- **Basados en expansión de diferencias:** Ventajas: se utiliza compresión sin pérdida y se puede cifrar el mapa de localización incrementando la seguridad. Desventajas: la capacidad de ocultamiento depende de las características de la imagen, entre más suave es la imagen mayor capacidad. Generalmente el mapa de localización es muy grande, aún con la compresión, así que disminuye notablemente la capacidad real de ocultamiento.
- **Basados en corrimiento al histograma:** Ventajas: es muy simple, ofrece valores de PSNR mayores a 48 dB y con gran capacidad de ocultamiento. Desventajas: la capacidad es limitada por la máxima frecuencia de los valores de intensidad de los píxeles.

Otros métodos no contemplados en [15]:

- **Basados en expansión del error de predicción** [23 y 27]: Ventajas: gran capacidad de ocultamiento con una pequeña distorsión de la

imagen. Desventajas: la necesidad de bits de bandera para poder recuperar la imagen original y la necesidad de un preprocesamiento, similar al proceso de ocultamiento, para reservar una región, garantizando que exista espacio para ocultar los bits extra generados en el proceso de ocultamiento.

- **Basado en corrimientos al histograma de las diferencias** [22]**:** Ventajas: tiene un control sencillo para la extracción de la marca y recuperar la imagen original. No necesita guardar algún mapa de localización o alguna otra información. El PSNR de la imagen marcada es mayor o igual a 51dB. Tiene adecuada capacidad de ocultamiento en imágenes con valores de píxeles homogéneos. Desventajas: los autores no sugieren como tratar el problema de desbordamiento (*underflow y overflow*), lo que ocasiona que en algunas imágenes no sea posible recuperar exactamente la imagen original debido a las pérdidas generadas por dicho desbordamiento.

Por otro lado, la mayoría de las publicaciones no presenta resultados con imagenología sensible y aún cuando teóricamente se puede tener alguna referencia de su comportamiento, es necesario poder tener un punto de comparación entre los diversos métodos con las imágenes de interés.

2.4 Implementaciones Software y Hardware

En cuanto a las implementaciones en software, los métodos publicados se pueden programar en cualquier lenguaje, pero también existen otras herramientas disponibles en Internet para ocultamiento de datos en cualquier tipo de archivo multimedia.

A continuación se presentan algunos ejemplos de herramientas en software libre para ocultamiento de datos.

- MP3Stego: oculta información en archivos MP3 durante el proceso de compresión. Los datos son primero comprimidos, cifrados y luego escondidos en el flujo de bits [9].
- JPHIDE y JPSEEK: son programas que permiten ocultar un archivo en una imagen jpeg [10].
- BlindSide Cryptographic Tool: esta herramienta puede ocultar archivos de cualquier tipo dentro de una imagen de mapa de bits de Windows (archivo BMP) [11].
- GIFShuffle**:** oculta mensajes con todas las imágenes GIF, incluyendo aquellas con transparencia y animación, y además proporciona compresión y cifrado de los mensajes ocultos [12].
- WbStego: es una herramienta que oculta cualquier tipo de archivo de mapa de bits en imágenes, archivos de texto, archivos HTML o archivos PDF de Adobe [13]
- MSU StegoVideo: permite ocultar cualquier archivo en una secuencia de video, pero genera una ligera pérdida de datos después de la compresión del video [14].

Adicionalmente a las herramientas de software se han propuesto plataformas hardware reconfigurables sobre FPGAs con el objetivo de acelerar el proceso de ocultamiento y recuperación de datos. Este tipo de plataformas se utilizan como una eficiente alternativa en aplicaciones de procesamiento digital de señales.

Se han publicado algunas implementaciones hardware para ocultamiento de datos, las cuales básicamente se enfocan en técnicas de marcas de agua en imágenes [34-35] y marcas de agua sobre video [36]. Pero no se han reportado implementaciones de algoritmos reversibles para ocultamiento de datos, en especial sobre imágenes.

2.5 FPGAs

Un FPGA (*Field-Programmable Gate Array*) es un dispositivo semiconductor que contiene componentes lógicos programables (bloques lógicos) y conexiones programables. Los bloques lógicos se pueden programar como compuertas lógicas o funciones combinacionales complejas de acuerdo a las necesidades específicas del cliente o del diseñador de hardware. En la mayoría de los casos los FPGAs incluyen elementos de memoria, los cuales van desde flip-flops hasta bloques de memoria más completos. La figura 6 muestra el esquema interno básico de un FPGA.

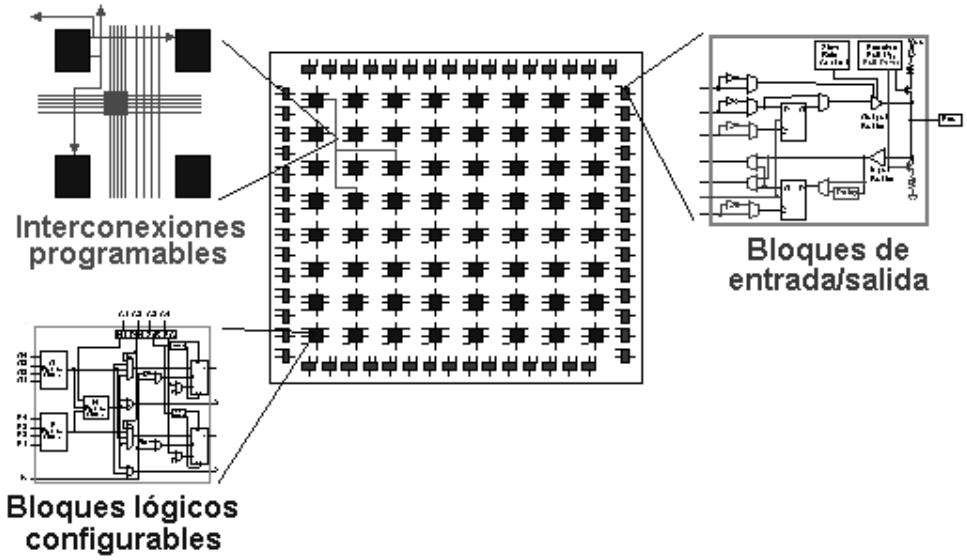


Figura 6: Esquema básico de un FPGA [34]

Los FPGAs generalmente son más lentos que los circuitos integrados de aplicación específica (*ASIC*) y no se recomiendan para diseñar dispositivos demasiado complejos o que requieren de una mayor cantidad de recursos o para dispositivos en fabricación en serie. Sin embargo, es más rápido diseñar las arquitecturas para los FPGAs ya que tienen la habilidad de poder reprogramarse después de hacer arreglos (depuración de errores).

Generalmente los diseñadores utilizan los FPGAs para diseñar y probar dispositivos y una vez teniendo el dispositivo final lo fabrican en serie en ASICs.

Para diseñar circuitos sobre dispositivos FPGAs se tiene una gran variedad de herramientas que facilitan el diseño. Se tienen tarjetas que además de contener al FPGA cuentan con interfaces a perifericos, dispositivos de memoria adicionales, incluso algunas cuentan con microprocesadores, etc. Las características de las familias de las tarjetas incluyendo las características del FPGA, varían de acuerdo a las diversas aplicaciones en las que pueden ser utilizadas. Entre las principales compañías líder en la fabricación y venta de tarjetas FPGAs se encuentran Xilinx y Altera.

Por su parte, Xilinx cuenta con una familia de tarjetas denominada Spartan 3E. Las plataformas Spartan 3E cubren las necesidades de gran volumen a bajo consumo y ofrecen un bajo costo total con soluciones de diseño completas para óptimos resultados [40].

Para el diseño y pruebas de la arquitectura del algoritmo propuesto se utilizó una tarjeta de este tipo.

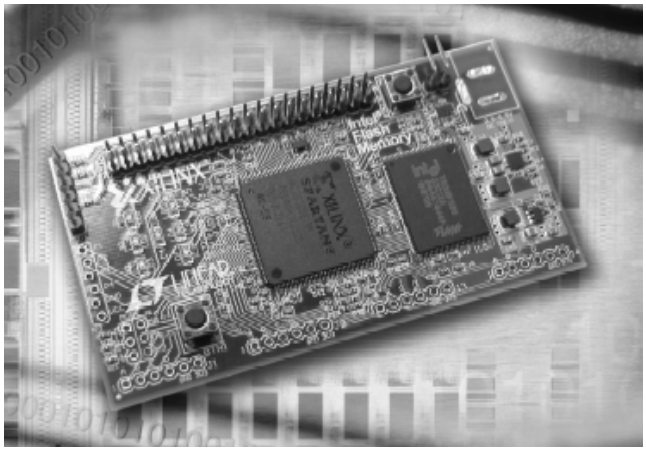


Figura 7: FPGA Spartan 3E de Xilinx

En resumen en este capítulo se proporcionó el marco teórico que sirve como base para este proyecto de tesis. Se introdujo a las principales técnicas de seguridad y especialmente se mencionaron los diversos métodos de ocultamiento de datos reversibles.

Así mismo se mencionó que mediante el uso de los FPGAs es posible diseñar arquitecturas hardware específicas mejorando el rendimiento ofrecido por las herramientas en software.

En el siguiente capítulo se abordarán los métodos de ocultamiento reversibles en los cuales se basa el método propuesto en el presente trabajo.

CAPÍTULO 3

MÉTODOS DE OCULTAMIENTO DE DATOS REVERSIBLES

Los primeros algoritmos publicados [18-21] han sido de gran influencia para los más recientes, donde los más recientes obtienen cada vez mejor rendimiento, en cuanto a mayor calidad en la imagen y mayor capacidad de ocultamiento. Siguiendo ese esquema de mejoramiento, se analizaron diferentes métodos de ocultamiento de datos del tipo reversible para así elegir a los que proporcionaran un mejor compromiso entre capacidad de ocultamiento, menor distorsión y con un control sencillo para recuperar la imagen original. Así se tomaron en cuenta las operaciones que realizan para el proceso de ocultamiento y recuperación de los datos y de la imagen original; la capacidad real de ocultamiento, es decir la capacidad de ocultamiento restándole los bits de control que requieren salvar para la exacta recuperación; y que al ocultar datos no generen distorsiones visibles a la imagen, principalmente. La capacidad de ocultamiento será medida en bits mientras que la distorsión de la imagen se evaluará con el PSNR (*Peak-Signal-to-Noise-Ratio*). Este valor está dado por la ecuación (1). Los valores

de PSNR mayores a 36 dB son aceptables en términos de degradación, lo que significa que no hay degradación significativa observada por el ojo humano [33].

$$PSNR = 10 \times \log_{10}\left(\frac{(ValorMax)^2}{MSE}\right) \quad (1)$$
$$ValorMax = (Bits - 1)^2$$
$$MSE = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}(I_O(i,j) - I_E(i,j))^2$$

Donde $ValorMax$ es el valor máximo representable; MSE (*Mean Squared Error*) es el error cuadrado medio, I_O es la imagen original e I_E es la estego imagen.

Los métodos analizados en general muestran buenos resultados pero con diversos compromisos, por ejemplo, generan muy poca distorsión a la imagen pero con el compromiso de obtener muy poca capacidad de ocultamiento [28, 32], otros generan una mayor capacidad de ocultamiento pero insertan ruido a la imagen, es decir mayor distorsión [29-31].

Así que se seleccionaron tres de los métodos cuyos resultados eran los más sobresalientes y cubrían los requisitos necesarios: mayor capacidad de ocultamiento con menor distorsión a la imagen y poniendo gran interés en el control utilizado. De estos tres métodos, se encontraron modificaciones por parte de otros autores que mejoraron su rendimiento, y en esas mejoras se ha basado el método propuesto.

Cada una de las siguientes subsecciones explica a detalle cada uno de los métodos elegidos. El título en cada una de las siguientes subsecciones hace referencia al autor principal del método explicado.

3.1 Método Lee

Este método fue propuesto por Sang-Kwang Lee y otros autores [22], ellos plantean una técnica de autenticación reversible para imágenes basada en marcas de agua. Su método es sencillo, con un tiempo de ejecución corto y produce poca distorsión a la imagen utilizando el histograma de las diferencias entre los valores de pares de píxeles adyacentes. Este método genera imágenes marcadas con valores de PSNR mayores a 51.14 dB. Para realizar la autenticación de la imagen, extraen los datos ocultos (la marca) de la imagen marcada y si se recupera exactamente la imagen original, entonces la imagen marcada es auténtica. A continuación se explicará el proceso de marcado de la imagen con este método y posteriormente el proceso de extracción de la marca y recuperación de la imagen original.

3.1.1 Marcado de la Imagen

Para marcar una imagen en escala de grises $I(i,j)$ de tamaño M x N píxeles, se debe formar una imagen de diferencias, $D(i,j)$ de tamaño M x $N/2$, se forma restando pares de píxeles adyacentes con (2), donde $I(i,2j+1)$ e $I(i,2j)$ son la columnas impares y las columnas pares, respectivamente.

$$D(i,j) = I(i,2j+1) - I(i,2j) \tag{2}$$

Una vez formada la imagen D, se efectuarán algunos corrimientos en su histograma de tal manera que los espacios de los valores -2 y 2 se liberen para el proceso de marcado. Es decir, si el valor de la diferencia es mayor o igual a 2, entonces se le suma 1 al píxel de la columna impar; si el valor de la

diferencia es menor o igual a -2 entonces se le resta 1 al píxel de la columna impar. Con esta modificación a D, es posible utilizar los valores de las diferencias igual a 2 y -2 para reconocer los píxeles que han sido marcados. Así la imagen modificada de las diferencias está dada por (3).

$$\tilde{D}(i,j) = \tilde{I}(i,2j+1) - I(i,2j) \tag{3}$$

Donde

$$\tilde{I}(i,2j+1) = \begin{cases} \mathrm{I}(i,2j+1)+1 & si \quad D(i,j) \geq 2 \\ I(i,2j+1)-1 & si \quad D(i,j) \leq -2 \\ I(i,2j+1) & otro \quad caso \end{cases} \tag{4}$$

Los autores de este método generan la marca que ocultarán en la imagen original mediante la combinación de una imagen binaria de algún logotipo junto el mensaje generado por una función hash, en este caso MD5. El algoritmo MD5 toma un mensaje de entrada de longitud arbitraria y produce como salida un arreglo de 128 bits. Se ha conjeturado que computacionalmente no es posible que una función hash genere dos mensajes distintos a partir del mismo mensaje de entrada, o que se pueda invertir la función para generar el mensaje de entrada. Sean

$$D_S = \left\{ D_s(i,j) : 0 \leq i \leq M-1; 0 \leq j \leq \frac{N}{2} - 1 \right\} \tag{5}$$
$$\tilde{D}_S = \left\{ \tilde{D}_s(i,j) : 0 \leq i \leq M-1; 0 \leq j \leq \frac{N}{2} - 1 \right\}$$

Entonces se calcula la hash *A(l)* de tamaño 128 como sigue:

$$Hs(K,M,N,\tilde{D}_s) = \{A(l) : 0 \leq l \leq 127\} \tag{6}$$

Donde K es la clave de usuario formada por una cadena de bits y $\tilde{D}_s$ es igual al correspondiente elemento en D_s exceptuando los valores de las diferencias igual a 1 y -1 que previamente se han puesto a 0. Para generar la marca de agua *W(m,n)* del mismo tamaño que la capacidad de ocultamiento C, se combina la hash *A(l)* con la imagen binaria del logotipo *B(m,n)* de tamaño P x Q píxeles usando la operación binaria XOR. En caso de que C > 128 y C > P x Q, se forma $\tilde{A}(l)$ y $\tilde{B}(m,n)$ replicando periódicamente *A(l)* y *B(m,n)* hasta el tamaño de C, respectivamente. Entonces la marca de agua para la autenticación se forma:

$$W(m,n) = \tilde{A}(l) \oplus \tilde{B}(m,n) \tag{7}$$

El siguiente proceso de modificación del histograma es para ocultar la marca de agua en la imagen. La imagen $\tilde{D}$ es escaneada y dependiendo del valor de la diferencia se ocultan o no bits en ese par de píxeles; esto es, si la diferencia es igual a -1 ó 1 entonces se oculta un bit de W, en este caso, si el bit a ocultar es '1' se resta o se suma 1 al píxel impar correspondiente, si el bit a ocultar es '0' entonces el píxel impar no se modifica, de esta manera cuando se oculta un '1' el valor de la diferencia cambia a -2 o 2, según el caso.

De esta manera las columnas impares de la imagen marcada, $I_w(i,2j+1)$, se obtienen como sigue:

Si *W(m,n)*=1 y $\tilde{D}(i,j)$=1 o -1,

$$I_w(i,2j+1) = \begin{cases} \tilde{I}(i,2j+1)+1 & si \quad \tilde{D}(i,j) = 1 \\ \tilde{I}(i,2j+1)-1 & si \quad \tilde{D}(i,j) = -1 \end{cases} \tag{8}$$

Y en otros casos,

$$I_w(i,2j+1) = \tilde{I}(i,2j+1) \tag{9}$$

Y las columnas pares de la imagen marcada:

$$I_w(i,2j) = \tilde{I}(i,2j) \tag{10}$$

El proceso de marcado de este método se muestra en la figura 8.

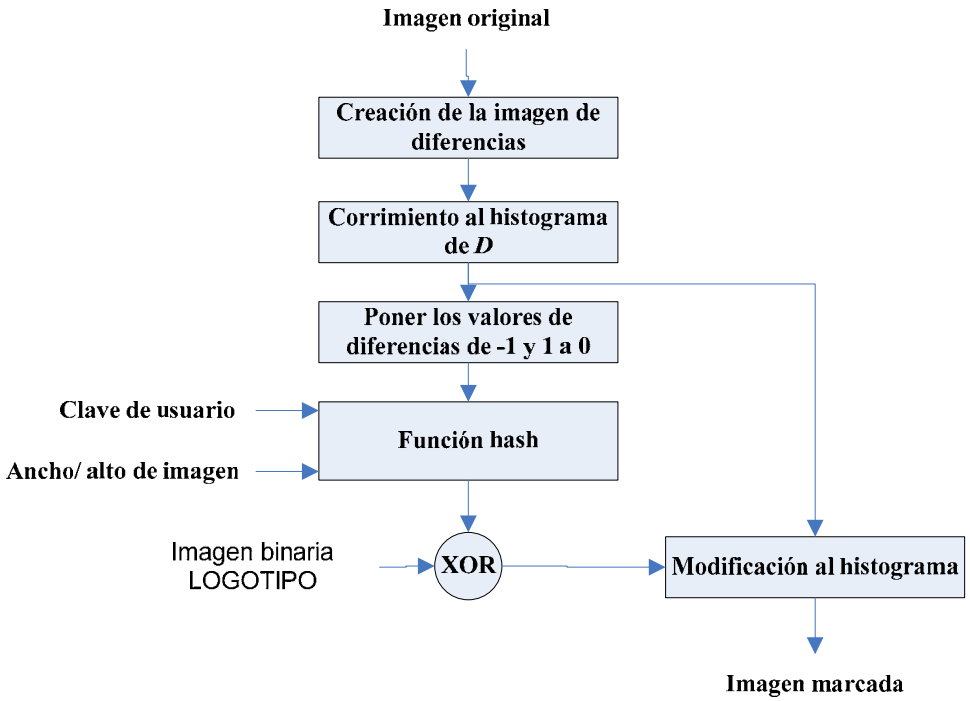


Figura 8. Proceso de marcado del método Lee

La capacidad de ocultamiento de este método está en función del número de pares de píxeles en la imagen cuyos valores difieran en -1 o 1. Este esquema se basa en el hecho de que un gran número de valores de las diferencias entre pares de píxeles tienden a estar alrededor de 0.

En cuanto a la distorsión de la imagen, en el peor de los casos a todos los píxeles de las columnas impares se le suma o resta 1; el MSE en este caso es 0.5. Por ello el PSNR de la imagen marcada está dado por (11), este valor está calculado suponiendo que la imagen tenga píxeles de 8 bits.

$$PSNR = (10\log_{10}(255^2 \times 2)) \approx 51.14dB \tag{11}$$

3.1.2 Extracción de Datos y Recuperación de la Imagen

En este proceso se verifica que la imagen no haya sido modificada y que sea auténtica y la imagen original se recupera a partir de la imagen marcada.

El proceso de extracción y recuperación es inverso al de marcado, primero se toma la imagen marcada recibida, $I_e(i,j)$ de tamaño M x N píxeles, se debe formar la imagen de diferencias, $D_e(i,j)$ de tamaño M x $N/2$, de la misma forma que en el proceso de marcado. Para la extracción de la marca $W_e(m,n)$ se deben buscar los pares de píxeles donde la diferencia sea -1,1,-2 o 2, y dependiendo del valor encontrado se extrae un '1' o '0' según (12).

$$W_e(m,n)=\begin{cases}0 & si \quad D_e(i,j)=-1 \, o \, 1\\ 1 & si \quad D_e(i,j)=-2 \, o \, 2\end{cases} \tag{12}$$

Al mismo tiempo se debe recuperar la imagen original invirtiendo los corrimientos realizados en el proceso de marcado, para las columnas impares según la ecuación (13) y las columnas pares están dadas por (14).

$$I_r(i,2j+1)=\begin{cases}I_e(i,2j+1)-1 & si \quad D_e(i,j)\geq 2\\ I_e(i,2j+1)+1 & si \quad D_e(i,j)\leq -2\\ I_e(i,2j+1) & otrocaso\end{cases} \tag{13}$$

$$I_r(i,2j)=I_e(i,2j) \tag{14}$$

Para extraer la marca de la imagen recibida es necesario utilizar la misma función hash obteniendo $\tilde{A}_e(k)$, se realiza con las mismas entradas que en el proceso de marcado; esto es, la imagen formada por las diferencias D_r, la imagen de las diferencias exceptuando los valores -2,-1,1 y 2 que se igualan a cero $\tilde{D}_r$.

$$D_r = \left\{ D_r(i,j) : 0 \le i \le M-1; \quad 0 \le j \le \frac{N}{2} - 1 \right\}$$
$$\tilde{D}_r = \left\{ \tilde{D}_r(i,j) : 0 \le i \le M-1; \quad 0 \le j \le \frac{N}{2} - 1 \right\} \tag{15}$$

$$Hs(K, M, N, \tilde{D}_r) = \left\{ \tilde{A}_r(k) : 0 \le k \le 127 \right\} \tag{16}$$

Posteriormente se realiza una operación XOR entre los valores $W_e(m,n)$ y $\tilde{A}_e(k)$ obtenidos. Al igual que en el proceso de marcado, si la capacidad de ocultamiento C>128 entonces se replican periódicamente los valores de $W_e(m,n)$ y $\tilde{A}_e(k)$.

$$\tilde{B}_e(m,n) = \tilde{A}_e(k) \oplus W_e(m,n) \tag{17}$$

Finalmente se forma $\tilde{B}(m,n)$ replicando periódicamente $B(m,n)$ hasta alcanzar el tamaño de C y se verifica la integridad de la imagen comparando $\tilde{B}(m,n)$ contra $\tilde{B}_e(m,n)$.

El proceso de extracción y recuperación de este método se muestra en la figura 9.

3.1.3 Ventajas y Desventajas

Este método tiene muchas ventajas sobre otros algoritmos, entre ellas se encuentran las siguientes: 1) Tiene un control sencillo para la extracción de la marca y recuperación de la imagen original, 2) no necesita guardar algún mapa de localización o alguna otra información, 3) el PSNR de la imagen marcada es mayor o igual a 51dB y 4) tiene adecuada capacidad de ocultamiento en imágenes con valores de píxeles homogéneos.

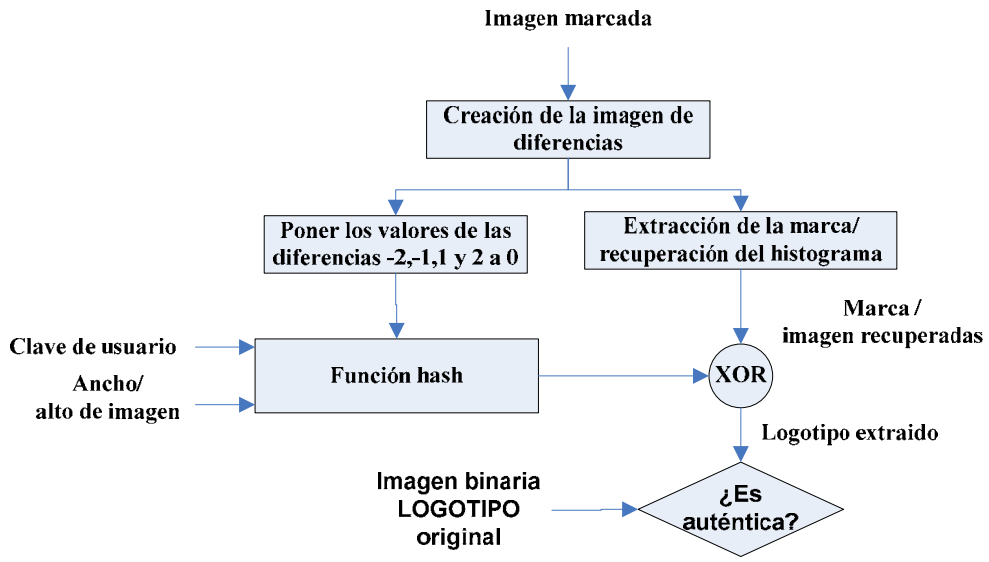


Figura 9. Proceso de extracción y recuperación del método Lee

Este método realiza corrimientos en los valores de los píxeles por al menos un valor, pero se genera un problema de desbordamiento cuando se realiza algún corrimiento a los valores de los píxeles que están cercanos a los límites de la escala de grises, lo que significa que los valores en los límites pudieran dar un salto hasta el otro límite del histograma, por ejemplo efectuar un corrimiento a la izquierda del valor 0, ocasionaría un desbordamiento cambiando el valor a 255 (si el píxel fuese representado por 8 bits). Los autores de este método no sugieren cómo tratar los posibles problemas de desbordamiento (*underflow y overflow*), lo que ocasiona que en algunas imágenes no sea posible recuperar exactamente la imagen original debido a las pérdidas de los valores originales generadas por dicho desbordamiento.

3.2 Método Ni

El método de Zhicheng Ni y otros autores [24] fue propuesto como una técnica de ocultamiento de datos reversible, la cual utiliza los *puntos cero* en el histograma de la imagen para ocultar los datos dentro de la misma. El punto cero es aquel valor en el histograma al cual no le corresponde ningún píxel de la imagen; es decir, el valor en el histograma igual a cero.

Es un algoritmo sencillo y no genera distorsión visible en la imagen modificada, el PSNR de la imagen original contra la imagen marcada es mayor a 48 dB.

3.2.1 Ocultamiento de Datos en la Imagen

Para una imagen en escala de grises de tamaño MxN, se genera primero su histograma $H(x)$. En el histograma se deben encontrar el *punto cero* y el *punto pico*. El *punto cero* corresponde al valor en la escala de grises tal que ningún píxel en la imagen asume dicho valor o en su defecto, al valor mínimo del histograma. El *punto pico* corresponde al valor en la escala de grises que tiene el máximo número de píxeles en la imagen. El objetivo de encontrar el *punto pico* es aumentar la capacidad de ocultamiento en la imagen, ya que el número de bits que pueden ocultarse en la imagen es igual al número de píxeles asociados al *punto pico*.

En algunas imágenes es posible que no exista el *punto cero* en el histograma. Para estos casos, se tomará el *punto mínimo*; esto es, el valor en la escala de grises que tenga el mínimo número de píxeles en la imagen. Generalizando estos términos nos referiremos al *punto cero* o *mínimo* como *punto mínimo* y el *punto pico* como *punto máximo*.

Sean el punto máximo *h(a)* y el punto mínimo *h(b)*, con a y $b \in [0,255]$; entonces si *h(b)*>0, se guardarán las coordenadas de los píxeles que tengan ese valor b, así como el valor mismo, entonces se pondrá *h(b)*=0.

Si $a<b$, es decir si el valor del píxel con *punto máximo es* mayor que el valor del píxel del *punto mínimo,* entonces se escanea la imagen y a los valores de los píxeles $x \in (a,b)$ se les hace un corrimiento a la derecha en *H(x)*, es decir, todos los píxeles con valores x entre (a,b) se les suma 1. Esto causa un hueco en el valor de la escala de grises a+1. Entonces se escanea nuevamente la imagen y los píxeles con valor igual a a se le suma 1 si el correspondiente bit a ocultar es '1', en caso contrario no se altera el valor.

Pero si $a>b$ entonces se escanea la imagen y a los valores de los píxeles $x \in (a,b)$ se les hace un corrimiento a la izquierda en *H(x)*, es decir, a todos los píxeles con valores x entre (a,b) se les resta 1. Esto causa un hueco en el valor *a* en la escala de grises. Entonces se escanea nuevamente la imagen y a los píxeles con valor igual a a-1 se les suma 1 si el correspondiente bit a ocultar es '1', en caso contrario no se altera el valor.

3.2.2 Aumentando la Capacidad de Ocultamiento

La capacidad de ocultamiento C es igual al valor máximo del histograma *h(a)*, menos el espacio extra requerido para guardar el valor mínimo encontrado y en caso de que no existiera algún *punto cero*, se guardarían también las coordenadas de los píxeles con el *valor mínimo*. En caso de ser necesaria una mayor capacidad de ocultamiento, es posible utilizar múltiples pares de *puntos mínimos* y *máximos*. Así el proceso de ocultamiento sería de la siguiente manera.

Primero se genera el histograma de la imagen, se buscan los *k-puntos mínimos*, con $1 \le k \le K$, donde K es el máximo número de pares posibles

(a,b) no traslapados en la imagen. Suponiendo que los *puntos mínimos* encontrados son: $h(b_1), h(b_2)...h(b_k)$ y asumiendo que $0 < b_1 < b_2 < ... < b_k < 255$, se buscan los *k-puntos máximos* en los intervalos intermedios de los *puntos mínimos*, de tal manera que se encuentren *k*-pares de *mínimos* y *máximos* no traslapados. Una vez realizado este proceso, se sigue el algoritmo original para cada par (a,b).

Es importante guardar el número de pares (a,b) que se utilizaron, así como los *k*-valores de a y b y la información extra generada por no encontrar *puntos ceros*; esto para lograr la exacta recuperación de la imagen original.

El proceso de ocultamiento se muestra en la figura 10.

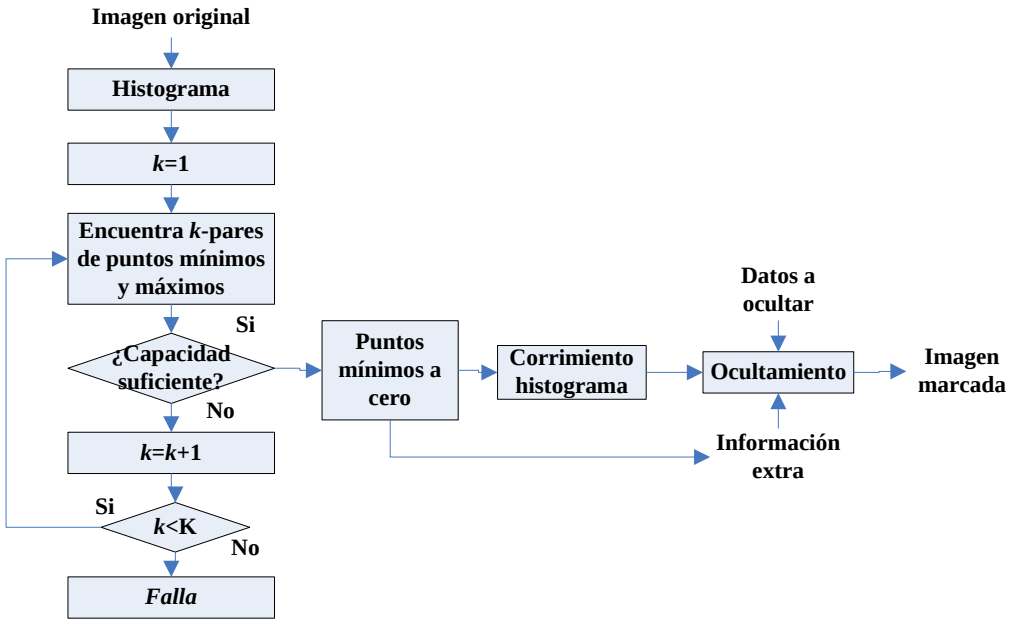


Figura 10. Proceso de ocultamiento del método Ni

Realizando este proceso de ocultamiento con *k*-pares de mínimos y máximos se puede generar una mayor capacidad de ocultamiento que con el algoritmo original, siendo la capacidad de ocultamiento igual a la suma de los *k-valores máximos* menos la información extra que se deba guardar,

incluyendo los valores mínimos y máximos y las coordenadas de los píxeles de la imagen con cada valor mínimo.

Sin embargo, al usar *k-pares* de *mínimos y máximos* la calidad de la imagen sería menor que si sólo se usara un par. En cuanto a la distorsión de la imagen, en el peor de los casos a todos los píxeles entre cada par de *mínimos* y *máximos* se le suma o resta 1; el MSE en este caso es máximo 1. Por ello el PSNR de la imagen marcada está dado por (18), este valor está calculado suponiendo que la imagen tenga píxeles de 8 bits.

$$PSNR(dB) = 10\log_{10}(255^2 / MSE) \approx 48.13 \qquad (18)$$

3.2.3 Extracción de Datos y Recuperación de la Imagen

Para realizar este proceso, es necesaria la información extra almacenada en el proceso de ocultamiento, tal como el valor de k, el valor de los *k-puntos mínimos* y *máximos* y las coordenadas de los píxeles que tienen el valor de cada *punto mínimo*. Una vez teniendo estos valores, para cada par de mínimos y máximos, se efectuará el siguiente proceso de extracción y recuperación.

Sean a y b el valor del punto *máximo* y *mínimo* respectivamente. Si $a<b$ entonces se escanea la imagen y los píxeles con valor a indican que el bit oculto es '0' y el valor del píxel no necesita ser modificado. Sin embargo, si el valor del píxel es igual a $a+1$ entonces el bit oculto es '1' y el valor del píxel se reduce en 1. Todos los valores de los píxeles con valores entre $a+2$ y b se reducen también en 1.

Pero si $a>b$ entonces se escanea la imagen y los píxeles con valor a indican que el bit oculto es '1' y el valor del píxel no necesita ser modificado. Sin embargo, si el valor del píxel es igual a $a-1$ entonces el bit oculto es '0' y

el valor del píxel se incrementa en 1. Todos los valores de los píxeles con valores entre b y a-2 se incrementan también en 1.

Después de recorrer todos los valores entre los *k-pares* de a,b y de recuperar los datos ocultos, es necesario recuperar los valores de los píxeles de la imagen original cuyas coordenadas fueron guardadas antes del proceso de ocultamiento por causa de no encontrar suficientes *puntos ceros* en el histograma. La figura 11 muestra el proceso de recuperación y extracción para cada par de *mínimos y máximos*.

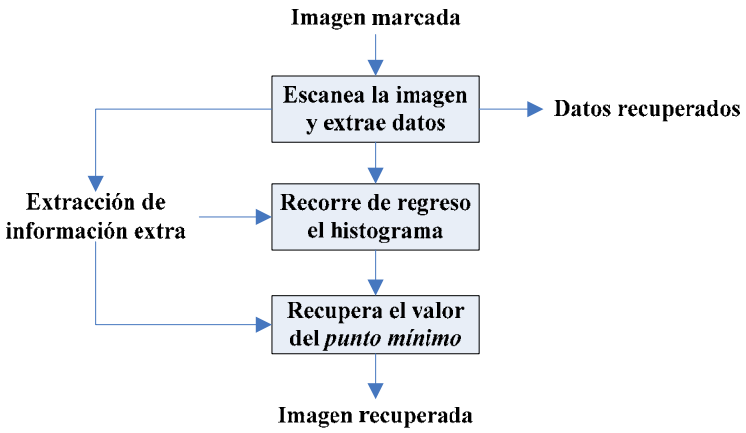


Figura 11. Proceso de extracción y recuperación del método Ni

3.2.4 Ventajas y Desventajas

Este método utiliza un procesamiento sencillo, básicamente escanea la imagen completa tres veces en total, obtiene el histograma de la imagen, encuentra los puntos mínimos y máximos y realiza los corrimientos necesarios para generar huecos y así ocultar los datos; utiliza un tiempo corto para procesar la imagen sin generar distorsiones visibles.

La gran desventaja en comparación con el método anterior, es la necesidad de guardar información extra para la recuperación exacta de los datos ocultos y de la imagen.

3.2.5 Mejoras Propuestas a este Método

Los autores M. Fallahpour y M. H. Sedaaghi [25] propusieron que para incrementar el rendimiento en cuanto a la capacidad de ocultamiento en la imagen, se podría realizar el procedimiento por bloques. Esto es, la imagen original se divide en *n-bloques* no traslapados, y posteriormente se efectúa el proceso de ocultamiento empleando *k-pares* de *mínimos* y *máximos* sobre cada bloque individualmente.

Junto con la información extra que se debe guardar, es necesario guardar también el valor de *n*.

3.3 Método Thodi

Diljith M. Thodi y Jeffrey J. Rodriguez [23] propusieron un algoritmo reversible de marcas de agua para imágenes digitales que oculta información en el error de predicción entre píxeles adyacentes usando un algoritmo para predecir el límite medio llamado *MED Predictor* (*Medium Edge Predictor*) tomado de [19] modificado para un contexto anti-causal, es decir que no depende de valores modificados anteriormente. El contexto de un píxel x para este caso son sus píxeles vecinos: el píxel de abajo, de la derecha y de abajo a la derecha (a,b y c respectivamente). El valor predicho $\hat{x}$ y el error de predicción p_e están dados por (19).

$$\hat{x} = \begin{cases} \max(a,b) & si \quad c \le \min(a,b) \\ \min(a,b) & si \quad c \ge \max(a,b) \\ a+b-c & en\ otro\ caso \end{cases} \tag{19}$$

$$p_e = x - \hat{x}$$

El píxel de la última fila y la última columna tienen un solo píxel como contexto; entonces su error de predicción se iguala al valor del píxel referenciado.

Para ocultar datos dentro de la imagen se va generando el error de predicción para cada píxel a medida que la imagen es escaneada, para ocultar un bit i en x se representa al error de predicción de forma binaria, se le hace un corrimiento a la izquierda y en el LSB vacante se inserta i. Si l es el tamaño de la representación binaria de p_e, entonces el valor modificado de p_e se representa en (20).

$$\begin{aligned} p_e &= b_{l-1}b_{l-2}...b_0 \\ \hat{p}_e &= b_{l-1}b_{l-2}...b_0 i = 2\,p_e + i \end{aligned} \tag{20}$$

Posteriormente se modifica el valor del píxel x,

$$x' = \hat{x} + \hat{p}_e = x + p_e + i \tag{21}$$

Es posible recuperar el valor original de x invirtiendo este proceso (22) lo cual se tendrá que realizar en el proceso de extracción y recuperación.

$$\begin{aligned} p_e &= \lfloor \hat{p}_e / 2 \rfloor \\ x &= x' - p_e - i \end{aligned} \tag{22}$$

3.3.1 Ocultamiento de Datos en la Imagen

Para ocultar datos en una imagen I de tamaño M x N y con L niveles de intensidad representables, primero es necesario realizar un preprocesamiento a la imagen para separarla en dos regiones R y S, la primera es una región reservada para garantizar que exista espacio disponible en la imagen para guardar la información extra que se genera para el proceso de recuperación y la segunda región será en donde se oculten los

datos. En este proceso se debe identificar el número de píxeles candidatos para la inserción de información, de manera que sea invertible; también se identifican los píxeles que pudieran ocultar datos pero que requieren de una bandera para su recuperación, así como los píxeles que no se utilizarán para ocultar pero que también requieren banderas. Estas banderas se irán insertando en el flujo de bits a ocultar, es por ello que se reserva una región para ocultar todos los bits extra que se van generando. La cantidad de bits extra determinan la cardinalidad de R, se reserva la región R del tamaño necesario y el resto de la imagen será la región S, en esta región S es donde se iniciará el proceso de inserción.

En este proceso se deberá guardar el bit menos significativo de cada píxel en R para poder recuperar exactamente esta región; se guardan en un arreglo de bits denominado R_{LSB}.

Terminado el preprocesamiento, se forma el flujo de datos a insertar B. Se forma concatenando el flujo de bits de la marca a ocultar P con un indicador de término EOP junto con el arreglo de bits R_{LSB}. Así $B = P \cup EOP \cup R_{LSB}$.

Para cada píxel en S, exceptuando la última fila y la última columna, se calculan los valores de error de predicción, el error de predicción p_e y el valor del píxel x determinan en dónde se ocultarán bits mediante la expansión del error de predicción. Así cuando p_e es menor que un determinado umbral T y después de insertar un bit no existe algún tipo de desbordamiento entonces esa localidad (el píxel referenciado x junto con su valor p_e) se usará para ocultar un bit i usando (20) y (21), estas localidades se denominan el conjunto E. El conjunto N_e serán todas las localidades donde $p_e > T$.

El conjunto $\mathrm{U_e}$ contiene todas las localidades en E y $\mathrm{N_e}$ que pueden causar ambigüedad en el decodificador, esto es, si el valor modificado del píxel $x' \in (0, T-1] \cup [L-1-T, L-1)$ se necesitarán bits de bandera *OFB´s*

(*Overflow Flag Bits*). El conjunto $U = S - E - N_e$, contiene todas las localidades que no pueden ser cambiadas y también requieren bits de bandera para ser reconocidos.

Dependiendo a que conjunto pertenece cada localidad, se ocultará un bit i de B, o se generará un bit de bandera de la siguiente manera:

- Si pertenece a E entonces se toma el primer bit de B y se inserta en esta localidad. Si también pertenece a U_e, se inserta el bit de bandera '1' al inicio de B.
- Si pertenece a N_e entonces se hace un corrimiento por T posiciones a la derecha cuando p_e ≥0 o un corrimiento a la izquierda por T-1 posiciones cuando p_e<0, de esta manera estas localidades se podrán distinguir entre las que se usaron para ocultar. Si también pertenece a U_e, se inserta el bit de bandera '1' al inicio de B.
- En otro caso, pertenece a U, se inserta el bit de bandera '0' al inicio de B.

Una vez terminado el proceso sobre la región S, si aún quedan bits sobrantes de B, entonces esos bits serán insertados en los bits menos significativos de los píxeles de la región R, la cual se reservó con un preprocesamiento garantizando así la capacidad suficiente para guardar los bits sobrantes de B.

El proceso de ocultamiento se muestra en la figura12.

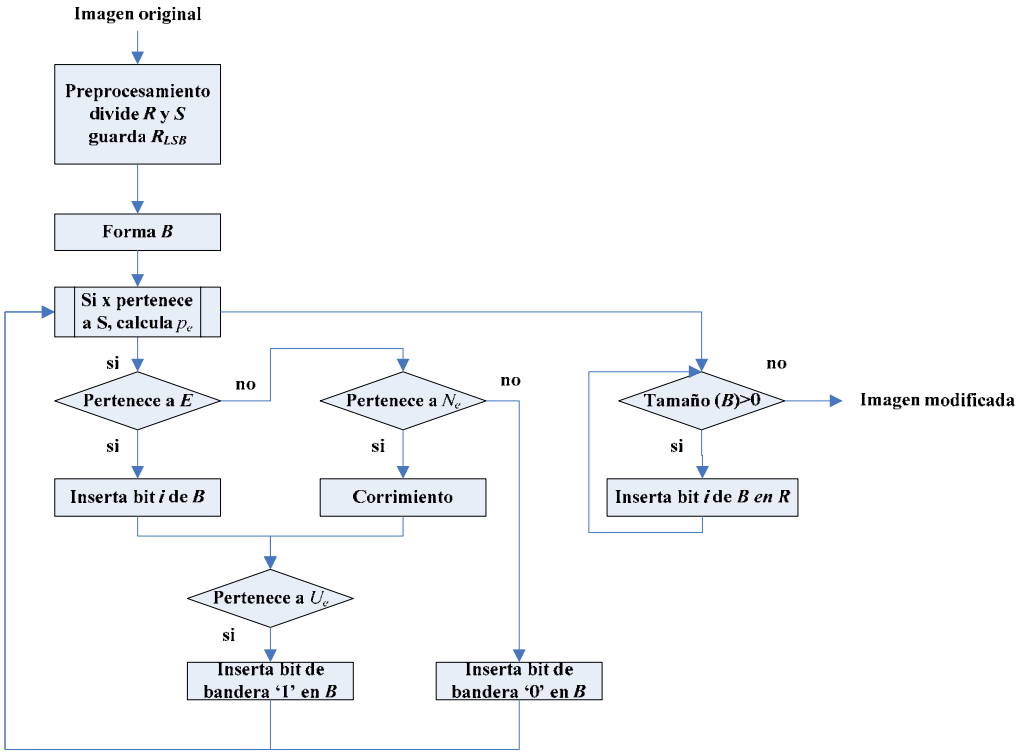


Figura 12. Proceso de ocultamiento del método Thodi

3.3.2 Extracción de Datos y Recuperación de la Imagen

En este proceso la imagen debe ser escaneada en orden inverso que el proceso de ocultamiento. El decodificador extrae el bit oculto y restaura el píxel original en cada localidad, por ello es importante restaurar el valor del píxel original en la localidad actual antes de proceder en la siguiente localidad.

Se extraen los datos sobre la región S empezando en orden inverso. Entonces para cada píxel x' se calcula el valor p_e y después se verifica a que conjunto pertenece:

- Si $x' \in U_e \cup U$, es decir si $x' \in (0, T-1] \cup [L-1-T, L-1)$, se debe diferenciar a que conjunto pertenece según el bit de bandera que

se extrae del inicio del flujo de bits extraído B'. Si B' está vacío entonces se obtiene esta bandera del LSB del inicio de la imagen. Si la localidad pertenece a U entonces es una localidad sin bits ocultos, si pertenece a U_e se procede como el siguiente paso.

- Si $x' \in E - U_e$, es decir si $x' \notin (0, T-1] \cup [L-1-T, L-1)$, y $p_e \in (-2T+1, 2T)$, entonces se extrae el bit oculto y se restaura el valor del píxel original.
- Si $x' \in N - U_e$, es decir si $x' \notin (0, T-1] \cup [L-1-T, L-1)$, y $p_e \notin (-2T+1, 2T)$, entonces sólo se restaura el valor original del píxel mediante un corrimiento, es decir a $x=x'-T$ si p_e es positivo o $x=x'+T-1$.

3.3.3 Ventajas y Desventajas

La ventaja de este método es la gran capacidad de ocultamiento que genera con una minima distorsión en la imagen marcada. Sin embargo, las desventajas son: la necesidad de bits de control para recuperar la imagen original y la necesidad de preprocesamiento, similar al proceso de ocultamiento, para reservar una región garantizando que exista espacio para ocultar los bits extra generados en el proceso de ocultamiento. Esto incrementa el tiempo de ejecución del algoritmo para imágenes grandes.

3.3.4 Mejoras Propuestas a este Método

Los autores de este método publicaron un artículo más reciente [20] en donde muestran ligeras modificaciones a su propio método. Las modificaciones que muestran son en cuanto al tipo de control necesario para la recuperación de la imagen. También presentan una comparación de su método contra el método de Tian, y contra algunas modificaciones de Tian.

Los resultados publicados indican que el método propuesto inicialmente, llamado Thodi en este documento, es el que tiene mejor rendimiento, con mayor calidad en la imagen con los datos ocultos y con gran capacidad de ocultamiento.

3.3.5 Implementación Software

Se realizó la programación de estos tres métodos para probarlos con imágenes radiológicas, ya que no todas las publicaciones presentaban resultados con este tipo de imágenes, y debido a la naturaleza sensible de las mismas es necesario que los algoritmos tengan buen rendimiento. Las imágenes utilizadas para las pruebas son en escala de grises en formato DICOM [37] y se ocultó el texto estándar *Alice*, obtenido del corpus Canterbury [38] y con un total de 149 KB.

En las tablas 1, 2 y 3 se muestran los resultados obtenidos en algunas imágenes. Las etiquetas representan a los diferentes métodos.

- **Lee**: representa los resultados con el método Lee.
- **Ni**: usando la mejora del método Ni con 4 bloques y el mayor número de pares mínimo y máximo en cada imagen.
- **Thodi**: representa los resultados con el método Thodi.

Los métodos fueron programados con Matlab 7.4 R2007a. En las tablas 1,2 y 3 se muestra la capacidad de ocultamiento medida en bits y los bits de control requeridos por cada método, la diferencia entre estos valores da la capacidad real de ocultamiento. Las filas de *overflow* y *underflow*, miden el número de píxeles en la imagen que tienen algún problema de desbordamiento después de haber ocultado los datos. También se muestra el tiempo de ejecución en cada imagen.

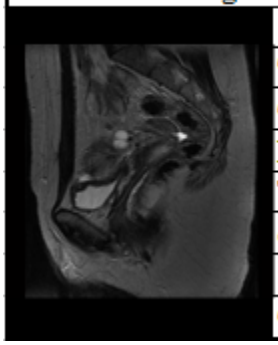
Imagen 1: MR-MONO2-12-an2.dcm 256x256x12			
	Lee	Ni	Thodi
Cap de Ocultamiento (bits)	467	8308	10309
Capacidad Real (bits)	467	7918	2467
PSNR (dB)	75.91	81.68	65.48
Tiempo de Ejecución (seg)	0.01	0.08	22.09
Overflow (píxeles)	0	0	0
Underflow (pixeles)	256	0	0
Control (bits)	0	390	7842

Tabla 1. Resultados de los métodos Lee, Ni y Thodi con la imagen 1

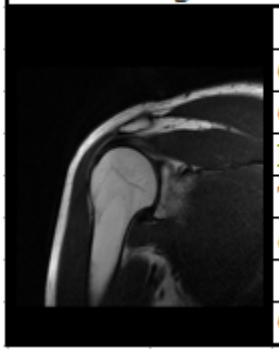
Imagen 2: MR-MONO2-12-shoulder.dcm 1024x1024x12			
	Lee	Ni	Thodi
Cap de Ocultamiento (bits)	22357	226762	270013
Capacidad Real (bits)	22357	226564	19367
PSNR (dB)	76.57	72.79	67.21
Tiempo de Ejecución (seg)	0.10	0.53	15222.36
Overflow (píxeles)	0	0	0
Underflow (pixeles)	2733	0	0
Control (bits)	0	198	250646

Tabla 2. Resultados de los métodos Lee, Ni y Thodi con la imagen 2

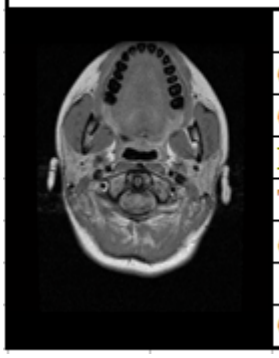
Imagen 3: brain_001.dcm 256x256x16			
	Lee	Ni	Thodi
Cap de Ocultamiento (bits)	1203	16291	18114
Capacidad Real (bits)	1203	15901	1074
PSNR (dB)	100.80	98.31	91.44
Tiempo de Ejecución (seg)	0.01	0.23	55.04
Overflow (pixeles)	0	0	0
Underflow (pixeles)	254	0	0
Control (bits)	0	390	17040

Tabla 3. Resultados de los métodos Lee, Ni y Thodi con la imagen 3

Con estos resultados es posible confirmar las ventajas de cada método, así como las desventajas que se hacen más notables. Lee no requiere de

bits extra para el control y tiene buenos niveles de PSNR aunque muestra poca capacidad de ocultamiento, es necesario tratar de alguna manera el problema del desbordamiento. Por otro lado Ni tiene el mejor rendimiento en cuanto a distorsión sobre capacidad de ocultamiento, requiere de pocos bits de control aunque pudieran aumentar si se cambian los parámetros usados en este método. Por último, aunque Thodi genera la mayor capacidad de ocultamiento, aun con sus mejoras, requiere de un control que almacena gran cantidad de bits extra lo que disminuye notablemente la capacidad real de ocultamiento y a medida que aumenta el tamaño de la imagen el procesamiento se vuelve más lento.

Con base en el estudio realizado, se sostiene que es necesario utilizar un método nuevo basado en las ventajas de los otros métodos generando una sinergia entre ellos.

CAPÍTULO 4

MÉTODO PROPUESTO

El método propuesto utiliza las principales ventajas de los métodos examinados y las características de las imágenes médicas. Tiene el objetivo de obtener gran capacidad de ocultamiento con una mínima distorsión de la imagen. Además utiliza un control sencillo para la exacta recuperación de la imagen original.

Se observó que el método de Thodi basado en la expansión del error de predicción tiene gran capacidad de ocultamiento, pero el control que requiere ocasiona que decrezca drásticamente esa capacidad. Por otro lado, el método de Lee no requiere de un control para la recuperación de la imagen original, pero la capacidad de ocultamiento es más baja que los otros métodos. Ni realiza un corrimiento al histograma para generar huecos en la imagen, y obtiene buenos resultados. Sin embargo, ya se han hecho mejoras a este método y es posible que no se pueda explotar más para obtener un mejor rendimiento. Por ello se utilizará como base un proceso de ocultamiento similar a Lee, pero buscando los valores más cercanos mediante un predictor, similar a Thodi. Para resolver el problema del desbordamiento que Lee no trata, se hará un preprocesamiento a la imagen

basado en un corrimiento al histograma similar al método Ni. En esta etapa de preprocesamiento, mediante un corrimiento al histograma de la imagen se podrán eliminar los píxeles que tuvieran problemas de desbordamiento después de ocultar bits en esas localidades.

A continuación se explica a detalle el método básico propuesto, iniciando por el proceso de ocultamiento de datos en la imagen, incluyendo el preprocesamiento ya mencionado; seguido de la explicación del proceso de extracción de los datos ocultos y la recuperación de la imagen original; por último se dará un ejemplo.

4.1 Ocultamiento de Datos en la Imagen

Como se ha mencionado, en este proceso se utiliza la expansión en el error de predicción para ocultar los datos en la imagen, modificando ligeramente el valor de los píxeles originales. Ante estas modificaciones se debe manipular cualquier problema de desbordamiento que pudiera surgir. De ahí que se utiliza un preprocesamiento en la imagen. En este preprocesamiento se efectúa un corrimiento al histograma de la imagen, estrechándolo de tal manera que los valores de los píxeles no estén en los límites de la escala de grises y al ocultar los datos no se genere un desbordamiento.

Primero se obtiene el histograma de la imagen y se encuentran dos puntos mínimos más cercanos a los límites del histograma (Z_{ini} y Z_{fin}), similar al método Ni y después se estrecha el histograma recorriendo los valores cercanos a los límites hacia Z_{ini} y Z_{fin}.

Para poder efectuar el corrimiento de regreso y recuperar la imagen original, se guarda un mapa (*mapa cero*) con los valores de Z_{ini} y Z_{fin}. El preprocesamiento se explica a más detalle en las siguientes subsecciones.

Una vez realizado el preprocesamiento sobre la imagen original I, se inicia el proceso de ocultamiento de la secuencia de datos B en la imagen preprocesada, llamada I_E; resultando la imagen modificada I_S con las mismas dimensiones que la imagen original. La última fila y la última columna no serán utilizadas para propósito de ocultamiento; esos píxeles permanecerán sin modificaciones para la recuperación exacta de la imagen original. Este proceso sigue los siguientes pasos:

I Preprocesamiento. Elimina posibles problemas de desbordamiento, este paso se explica a detalle en la siguiente subsección.

II En un determinado orden (de izquierda a derecha y de arriba a abajo), se escanea la imagen $I(i,j)$, donde $0 \le i \le M-1$ y $0 \le j \le N-1$

- Si $i=M$-1 ó $j=N$-1 entonces $I_S(i,j)= I_E(i,j)$
- si no, obtener el error de predicción p_e de $I_E(i,j)$ y su contexto, mostrado en (18)
  - Si p_e= 1 entonces ocultar el bit b del flujo de bits B, esto es:

$$I_S(i,j)= I_E(i,j) + b$$

  - Si p_e = -1 entonces ocultar el bit b del flujo de bits B, esto es:

$$I_S(i,j)= I_E(i,j) - b$$

  - Si p_e ≥2 entonces hacer un corrimiento a la derecha, esto es:

$$I_S(i,j)= I_E(i,j) + 1$$

  - Si p_e ≤-2 entonces hacer un corrimiento a la izquierda, esto es:

$$I_S(i,j)= I_E(i,j) - 1$$

  - En otro caso:

$$I_S(i,j)= I_E(i,j)$$

La figura 13 muestra el proceso de ocultamiento.

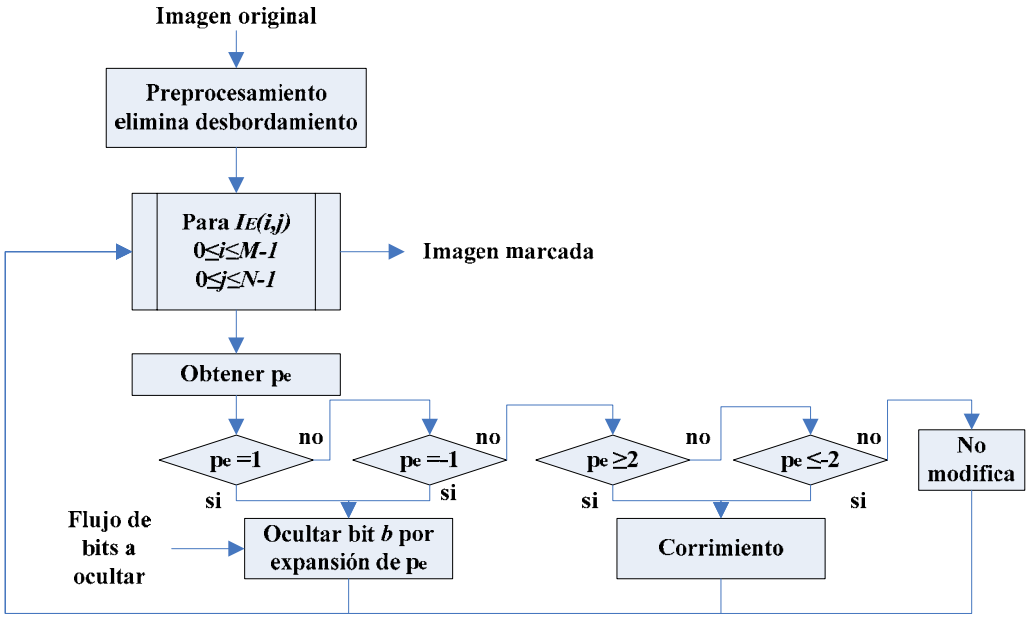


Figura 13. Proceso de ocultamiento del método propuesto

4.1.1 Preprocesamiento de la Imagen

En este paso se estrecha el histograma de la imagen de tal manera que no queden valores de píxeles en los límites de la escala de grises.

Para preprocesar una imagen en escala de grises $I(i,j)$ de tamaño M x N píxeles se siguen los siguientes pasos:

1. Se obtiene el histograma de los valores de los píxeles de la imagen.
2. Se encuentra el primero y el último *punto mínimo* del histograma Z_{ini} y Z_{fin}, si no se encuentra un valor *cero*, entonces se utilizará el valor *mínimo* guardando estos valores y la localización de los mismos en la imagen para su exacta recuperación; véase el método Ni.
3. Se realiza el estrechamiento, con cada píxel en la imagen:
   a. Si $I(i,j) < Z_{ini}$; se hace un corrimiento a la derecha, esto es:

$$I(i,j)=I(i,j)+1$$

b. Si $I(i,j) > Z_{fin}$; se hace un corrimiento a la izquierda, esto es:

$$I(i,j)=I(i,j)-1$$

c. En otro caso, no se modifica.

4. Se forma la secuencia de bits B, concatenando el *mapa cero* (ZM) con el flujo de datos a ocultar (P) y su respectivo indicador de término (EOP):

$$B = ZM \cup P \cup EOP$$

4.1.2 El Mapa Cero

El *mapa cero* ZM, figura 14, se necesita en el proceso de recuperación final. Se concatena el ZM al inicio del flujo de datos que será ocultado. En el caso de que el histograma no tenga valores *cero* entonces se tomará el valor *mínimo.* Los dos primeros bits del mapa serán ocupados para identificar si se encontraron puntos *cero* o *mínimos*, dependiendo de esos valores se construirá el resto del mapa. Los siguientes dos bits representan el ancho de los píxeles de la imagen (*BitDepth* = 8, 10, 12 ó 16 bits por píxel) y los siguientes bits serán usados por los valores de los puntos mínimos Z_{ini} y Z_{fin}. Cuando los valores de los puntos mínimos, no son cero, entonces se guardan sus valores K_{ini} y K_{fin} y su localización en la imagen ($tam(i,j)$ x K_{ini}/K_{fin}), en este caso son las coordenadas i,j de cada píxel con el valor K; $tam(i,j)$ será el número de píxeles necesarios para representar la coordenada, dado por el ancho y largo de la imagen. Con estos datos se garantiza la recuperación exacta de la imagen original.

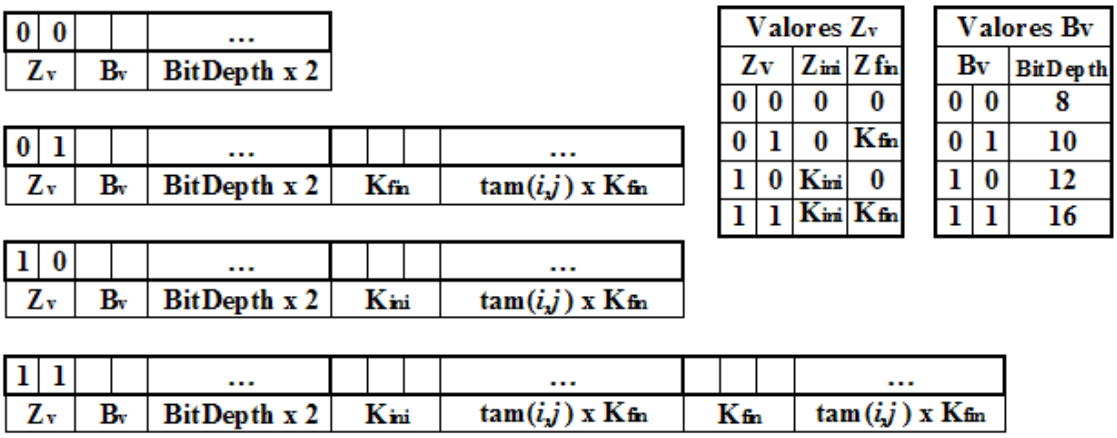


Figura 14. El *mapa cero* ZM

4.2 Extracción de Datos y Recuperación de la Imagen

El proceso de extracción y recuperación es inverso al proceso de ocultamiento. En este proceso se extraen los bits ocultos en la imagen en orden inverso a cuando fueron ocultados, es decir si en el proceso de ocultamiento el orden fue de izquierda a derecha y de arriba abajo entonces la extracción se realizará de abajo a arriba y de derecha a izquierda.

Se obtiene la imagen original I de I_S con los siguientes pasos:

1. Se escanea la imagen en orden inverso, y con cada píxel $I_S(i,j)$, (donde i =M-2 a 0 y j=N-2 a 0) :
    a. Obtener el valor del error de predicción p_e de $I_S(i,j)$ y su contexto, mostrado en (19),
        - Si p_e=1 ó p_e=-1 entonces el bit oculto b fue ‘0’. Concatenar ‘0’ al inicio de la secuencia de datos B_R y restaurar el valor del píxel $I_E(i,j)$=$I_S(i,j)$.

- Si p_e=-2 entonces el bit oculto b fue ‘1’. Concatenar ‘1’ al inicio de la secuencia de datos B_R y restaurar el valor del píxel $I_E(i,j)=I_S(i,j)$+1.
- Si p_e=2 entonces el bit oculto b fue ‘1’. Concatenar ‘1’ al inicio de la secuencia de datos B_R y restaurar el valor del píxel $I_E(i,j)=I_S(i,j)$-1.
- Si p_e>2 entonces realizar un corrimiento a la izquierda, esto es modificar el valor del píxel $I_E(i,j)=I_S(i,j)$-1.
- Si p_e<-2 entonces realizar un corrimiento a la derecha, esto es modificar el valor del píxel $I_E(i,j)=I_S(i,j)$+1.
- En otro caso el valor del píxel $I_E(i,j)=I_S(i,j)$.

Después de recuperar la secuencia de datos oculta B_R y la imagen I_E (es decir, la imagen que se tomó para ocultar los datos con el histograma estrecho), es necesario ahora decodificar el *mapa cero ZM* del inicio de B_R. Una vez obtenidos los valores de Z_{ini} y Z_{fin} y en caso de ser necesario, la localización de los píxeles con valores Z_{ini} y Z_{fin}, entonces se realiza el proceso de expansión del histograma para recuperar la imagen original *I*, de manera inversa al proceso de estrechar el histograma.

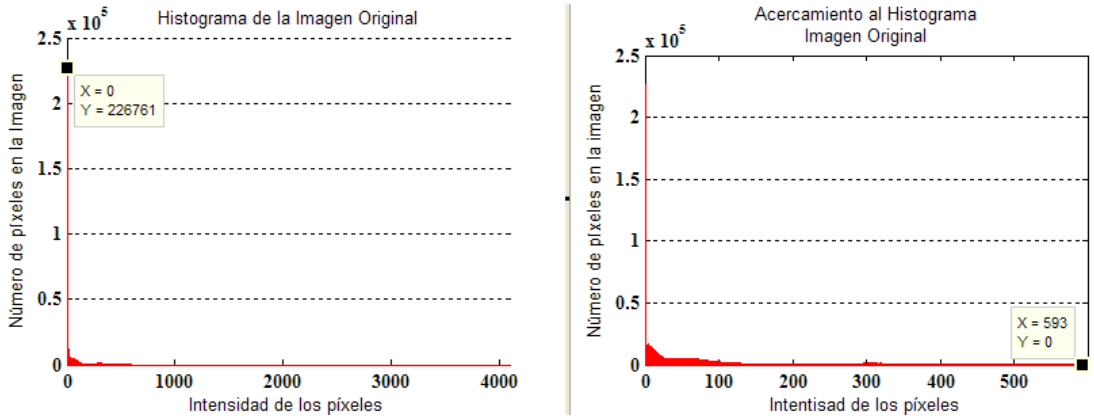
4.3 Ejemplo

Suponga que se requiere ocultar la secuencia de bits B= 11001010101… de *n*-bits en la imagen en formato DICOM: 1-MONO2-12-shoulder.dcm mostrada en la figura 15a. El primer paso es preprocesar la imagen. Se genera entonces su histograma, mostrado en la figura 15b. Podemos observar que 226761 píxeles tienen intensidad 0, es decir, un gran número de píxeles que dependiendo los valores del flujo de bits a ocultar,

ocasionarían problemas de desbordamiento. Por ejemplo, si se quisiera ocultar el texto Alice [38] se generarían 2733 píxeles con desbordamiento. Así que se realiza el estrechamiento al histograma según el algoritmo. El primer punto mínimo se encuentra en el valor 593 (véase la figura 15c).

Imagen: 1-MR-MONO2-12-shoulder.dcm

(a)



(b) **(c)**

Figura 15. Ejemplo de histograma

(a) Imagen original, (b) Histograma de la imagen original y (c) es un acercamiento del histograma con los primeros 600 valores

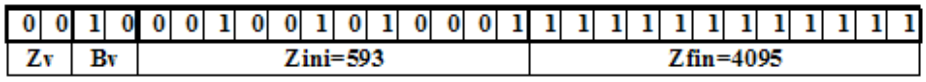
Entonces se realiza el corrimiento y se forma el ZM, mostrado en la figura 16a; en este caso los dos primeros bits '00' indican que Z_{ini} y Z_{fin} son puntos cero y que no necesitan guardar mapa de localización; los dos siguientes '10' indican que cada píxel está representado por 12 bits; los siguientes bits

representan el valor de Z_{ini} y Z_{fin} usados para el corrimiento. El histograma modificado después del corrimiento correspondiente, se muestra en la figura 16b. La imagen generada I_E, es la de la figura 16c.

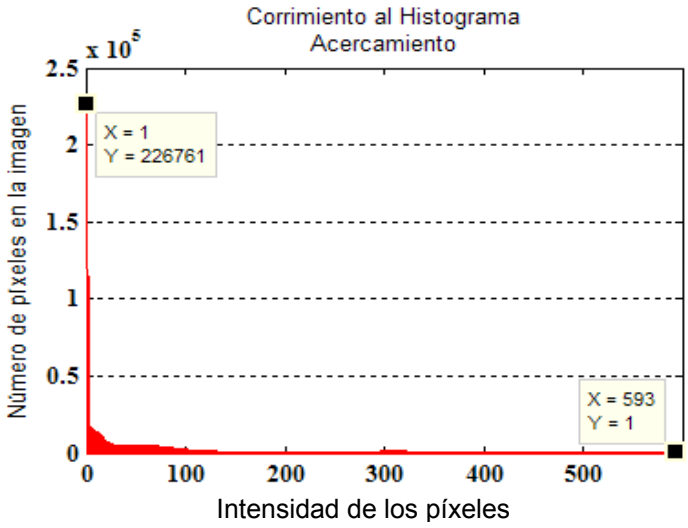
En la imagen I_E se ocultará la secuencia de bits. Para ilustrar el método de ocultamiento, suponga que nos encontramos en una región de la imagen de 18 píxeles, los cuales se muestran en la figura 17a; éstos no necesariamente corresponden a la imagen mostrada, mas bien son utilizados para efectos prácticos mostrando así todos los casos posibles.

Para cada píxel y su contexto se obtiene el valor predicho p (figura 17b), el error de predicción p_e (figura 17c), y dependiendo del bit b de la secuencia de bits B (figura 15d) se modifican los valores de los píxeles (figura 17e).

Sea x=1, y su contexto 2, 3 y 2, entonces el valor predicho p=2 y el error de predicción p_e=-1, el bit a ocultar b=1 entonces el valor $\hat{x} = x - b = 1 - 1 = 0$, y la secuencia de bits B= 100101010....



(a)



(b)

(c)

Figura 16: Ejemplo de corrimiento al histograma
(a) ZM para recuperar la imagen original, (b) corrimiento al histograma, los primeros 600 valores, (c) Imagen modificada I_E

El valor del siguiente píxel es x=3, su contexto 2, 2 y 2, con p=2, p_e=1 y b=1 entonces el nuevo valor es $\hat{x} = x + b = 3 + 1 = 4$ y la secuencia de bits queda B=00101010… y así sucesivamente, cuando p_e es mayor o igual a 2 entonces se realiza un corrimiento a la derecha, si p_e es menor o igual a -2 se realiza un corrimiento a la izquierda y si p_e es igual a cero, no se hacen cambios en el valor del píxel.

Se continúa el proceso hasta llegar al final de la imagen, sin modificar la última fila ni la última columna.

El proceso de extracción y recuperación se realiza en orden inverso. Es decir, se iniciará a extraer los bits ocultos desde el último píxel modificado. Siguiendo con nuestro ejemplo, el último píxel fue $\hat{x} = 21$ con p=20, p_e=1 entonces el bit oculto fue '0' y el valor del píxel no se modifica. El siguiente píxel sería $\hat{x} = 20$ con p=21, p_e=-1 entonces el bit oculto fue '0' y el valor del píxel no se modifica, así B_R='00'. El siguiente es $\hat{x} = 19$ con p=19, p_e=0 entonces no hay bit oculto, valor del píxel no se modifica y permanece B_R='00'. El siguiente es $\hat{x} = 19$ con p=15, p_e=4 entonces se realizó un corrimiento a la derecha, por tanto se realiza el proceso inverso y $x = \hat{x} - 1 = 19 - 1 = 18$, no hay bit oculto y permanece B_R='00'. El siguiente píxel sería $\hat{x} = 4$, pero ahora su contexto es 9,18 (valor ya modificado) y 15,

entonces p=12, p_e=-8, se realiza un corrimiento a la derecha y $x=5$ no hay bit oculto y permanece B_R=‘00’. Se sigue así sucesivamente hasta el inicio de la imagen, en nuestro caso en la región del ejemplo dado se recupera el flujo de bits ‘1100’, que fueron los bits ocultos y la región original.

Por último se debe realizar la expansión del histograma de la imagen I_E recuperada según los valores de ZM, recuperados al inicio de B_R.

I_E	1	3	2	5	18	19	20	21	20
	2	2	2	9	15	19	21	22	22

(a)

p	2	2	2	12	15	19	21	20

(b)

pe	-1	1	0	-7	3	0	-1	1

(c)

$$B = 1100101010...$$

bit oculto	1	1	No	<--	-->	No	0	0

(d)

I_S	0	4	2	4	19	19	20	21	20
	2	2	2	9	15	19	21	22	22

(e)

Figura 17: Ocultamiento de *B* en una región de la imagen

4.4 Método Propuesto para Imágenes Médicas Radiológicas

En este método utilizamos los valores del error de predicción iguales a 1 y -1 para ocultar los datos dentro de las imágenes; sin embargo se pueden utilizar las características específicas de las imágenes médicas radiológicas para obtener mayor capacidad de ocultamiento en este tipo de imágenes. Una característica es que las imágenes médicas tienen píxeles de intensidad regularmente homogénea. Otra característica es que el perímetro de la

imagen es comúnmente negro y la región de interés por parte del especialista es el centro de la imagen; así que si se modifica ligeramente este contorno no alteraría el diagnóstico médico.

Con estos antecedentes, se propone la ligera modificación de usar regiones negras de 2x2 píxeles de la imagen para ocultar mayor número de bits. Es decir, si el contexto de un referenciado píxel es una región negra y el referenciado píxel es también negro, entonces se ocultará un bit en ese píxel; si no, entonces se realizará un corrimiento a su valor a la derecha, vea la figura 16.

Este simple cambio podría incrementar significativamente la capacidad de ocultamiento en este tipo de imágenes; ya que dependiendo de la imagen radiológica, las regiones "negras" abarcan hasta aproximadamente el 85% de la imagen.

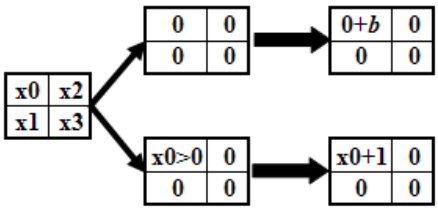


Figura 18. Modificación propuesta

Por otro lado, si se requiere una mayor capacidad de ocultamiento, es posible realizar el ocultamiento por capas. Podemos utilizar un segundo procesamiento sobre la imagen con bits ya ocultos, insertando una mayor cantidad de bits. Debido a que el método propuesto no distorsiona visiblemente la imagen original, es posible realizar una segunda vez el mismo proceso de ocultamiento con una ligera distorsión extra, aún imperceptible; véase la figura 19.

En el proceso de recuperación, primero se debe obtener el último flujo de bits oculto, de ahí se recupera una imagen intermedia, a la que se extrae el

primer flujo de bits oculto, recuperando así la imagen original. Finalmente se concatenan ambos flujos de bits recuperados, en el orden correcto obteniendo el total de los bits ocultos.

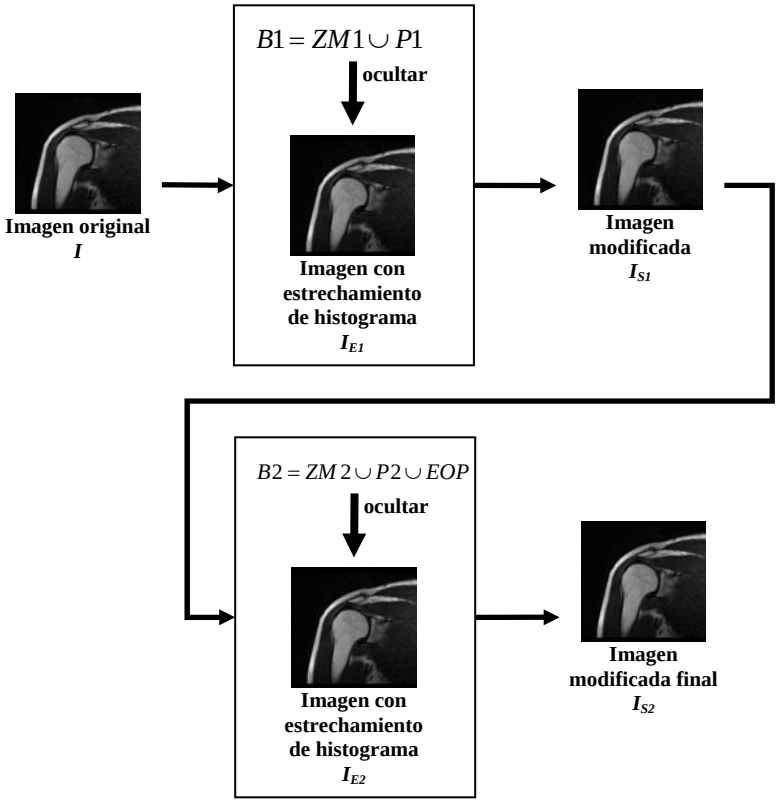


Figura 19. Proceso de ocultamiento en capas

En este capítulo se ha explicado a detalle el método propuesto, incluyendo algunas mejoras en cuanto a la capacidad de ocultamiento del mismo, especialmente en imágenes del tipo radiológicas.

En el siguiente capítulo se muestran las implementaciones realizadas del algoritmo propuesto y se analizan los resultados obtenidos al probarlas sobre algunas imágenes radiológicas. Los resultados de la implementación en software se comparan con los resultados obtenidos de los métodos analizados en el capítulo 3. Se muestra también la mejora en cuanto al tiempo de procesamiento de la implementación en hardware sobre la de software.

CAPÍTULO 5

IMPLEMENTACIONES Y RESULTADOS

Se programaron los métodos reversibles de ocultamiento de datos sobre los cuales se basó el método propuesto, también se programó el método propuesto con el fin de comparar los resultados de todos los métodos con un mismo banco de imágenes de prueba.

Así mismo se le hizo una pequeña modificación al método Lee basada en un preprocesamiento a la imagen, según el método propuesto en este documento, para eliminar los problemas de desbordamiento. Los resultados de esta modificación al método de Lee se incluyen en las tablas de comparación entre el método propuesto y los anteriores.

Los métodos se programaron con la herramienta Matlab 7.4 2007, se realizó en este intérprete por simplicidad de código y para revisar los parámetros de calidad de la imagen y capacidad de ocultamiento. También sirvió de base para verificar los tiempos de ejecución.

Las pruebas realizadas sobre el software de los diferentes métodos mencionados se realizaron con imágenes médicas radiológicas en formato

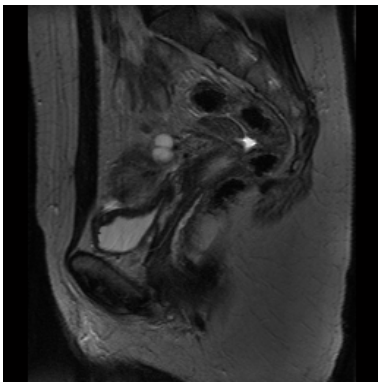
DICOM en escala de grises y de distintos tamaños. Se eligió este tipo de imágenes sensibles para verificar los parámetros de capacidad de ocultamiento, calidad de la imagen y el tiempo de ejecución del proceso de ocultamiento de cada método. Las imágenes se tomaron de algunos repositorios de imágenes médicas [37]. La figura 20 muestra las imágenes representativas cuyos resultados se muestran en esta sección.

El mensaje a ocultar es el texto estándar Alice del corpus Canterbury [38] con un total de 149 KB, usado generalmente para compresión de datos. Adicionalmente se realizaron pruebas ocultando otros textos obteniendo resultados similares. La estrategia de las pruebas fue intentar ocultar la mayor cantidad posible de bits en cada imagen.

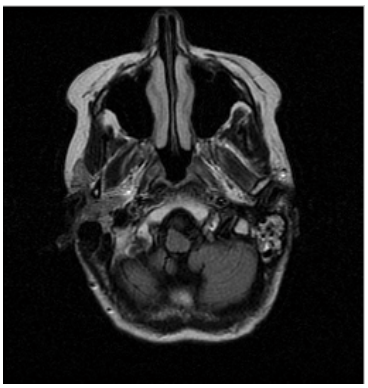
Debido a la tendencia creciente de sistemas embebidos y arquitecturas especializadas, en especial en el área médica, y dado que la complejidad del algoritmo es lineal hacen que el algoritmo propuesto sea atractivo para su implementación embebida en aplicaciones de bajo consumo de potencia. En este sentido se exploró una implementación hardware para mostrar la factibilidad en este tipo de sistemas.

El diseño se realizó con el lenguaje de descripción de hardware VHDL y con herramientas de Xilinx ISE 9.2i sobre una tarjeta FPGA Spartan 3E de Xilinx dispositivo xc3s500e-5-fg320. Para probar el funcionamiento de la arquitectura hardware se realizaron pruebas en simulación y con técnicas de hardware-en-el-ciclo (*hardware-in-the-loop)* con la herramienta de Simulink/Matlab. *Hardware-en-el-ciclo* es un paso que se utiliza comúnmente después de generar y compilar código de un modelo o implementación hardware en el que se descarga el ejecutable compilado a una computadora para correr esa aplicación.

Adicionalmente el método propuesto se programó en lenguaje C para comparar los resultados en cuanto a tiempo de procesamiento del método en software y hardware.

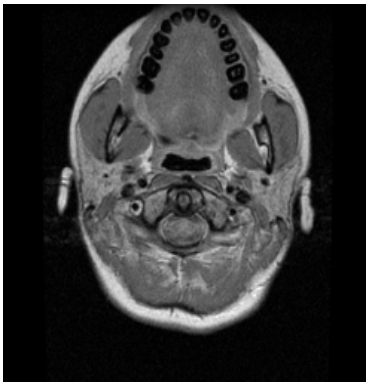


(a)

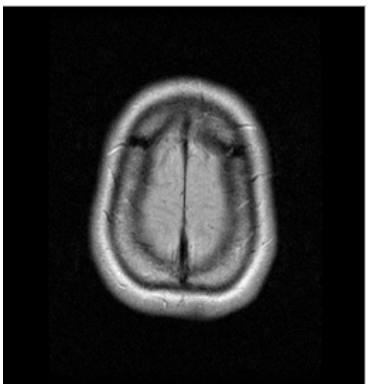


(b)

(c)



(d)



(e)

Figura 20: Imágenes de prueba.
(a) Imagen 1: MR-MONO2-12-an2.dcm 256x256x12;
(b) Imagen 2: MR-MONO2-16-head.dcm 256x256x16; (c) Imagen 3: MR-MONO2-12-shoulder.dcm 1024x1024x12 ;(d) Imagen 4: brain_001.dcm 256x256x16;
(e) Imagen 5: brain_020.dcm 256x256x16

5.1 Implementación Software

Los apéndices A1-A9 contienen el código fuente de las implementaciones de los métodos, tanto el propuesto como los que sirvieron de base.

Las tablas 4 a 8 muestran los resultados obtenidos con las diferentes imágenes. Las etiquetas representan a los diferentes métodos.

- Lee: representa los resultados con el método Lee.
- Ni: utilizando la mejora del método Ni, con 4 bloques y el máximo de pares de mínimos y máximos en cada imagen.
- Thodi: Usando el método Thodi con un umbral *T*=3.
- Lee+Sh: son los resultados de la mejora propuesta de Lee, realiza un preprocesamiento a la imagen de manera que elimina los píxeles con posibles problemas de desbordamiento.
- P1: es el algoritmo básico propuesto.
- P2: es el algoritmo propuesto para imágenes médicas radiológicas. Oculta datos en regiones negras de 2x2 (véase sección 4.4).
- P3: es la propuesta P2 añadiendo el ocultamiento en 2 capas. Se insertan los datos a ocultar en la imagen original y después se toma la estego-imagen y se toma como un nuevo medio, ocultando datos nuevamente sobre la imagen (véase sección 4.4).

Imagen 1: MR-MONO2-12-an2.dcm 256x256x12							
Método	Capacidad de Ocultamiento (bits)	Capacidad Real (bits)	PSNR (dB)	Tiempo de Ejecución (seg)	*Overflow* (pixeles)	*Underflow* (píxeles)	Control (bits)
Lee	467	467	75.91	0.01	0	256	0
Ni	8308	7918	81.68	0.08	0	0	390
Thodi	10309	2467	65.48	22.09	0	0	7842
Lee+Sh	467	439	74.85	0.02	0	0	28
P1	1112	1084	72.36	0.07	0	0	28
P2	8762	8734	72.23	0.07	0	0	28
P3	13572	13514	65.99	0.15	0	0	58

Tabla 4. Resultados del proceso de ocultamiento en la imagen 1

Imagen 2: MR-MONO2-16-head.dcm 256x256x16							
Método	Capacidad de Ocultamiento (bits)	Capacidad Real (bits)	PSNR (dB)	Tiempo de Ejecución (seg)	*Overflow* (pixeles)	*Underflow* (píxeles)	Control (bits)
Lee	2685	2685	99.85	0.01	0	961	0
Ni	2968	2706	97.12	0.23	0	0	262
Thodi	16044	10030	89.58	19.70	0	0	6014
Lee+Sh	2685	2649	94.73	0.03	0	0	36
P1	5887	5851	93.74	1.47	0	0	36
P2	6175	6139	96.62	1.81	0	0	36
P3	10625	10515	87.67	3.62	0	0	110

Tabla 5. Resultados del proceso de ocultamiento en la imagen 2

Imagen 3: MR-MONO2-12-shoulder.dcm 1024x1024x12							
Método	Capacidad de Ocultamiento (bits)	Capacidad Real (bits)	PSNR (dB)	Tiempo de Ejecución (seg)	*Overflow* (pixeles)	*Underflow* (píxeles)	Control (bits)
Lee	22357	22357	76.57	0.10	0	2733	0
Ni	226762	226564	72.79	0.53	0	0	198
Thodi	270013	19367	67.21	15222.36	0	0	250646
Lee+Sh	22357	22329	70.87	0.47	0	0	28
P1	62707	62679	69.95	1.21	0	0	28
P2	278482	278454	69.76	1.32	0	0	28
P3	453165	453107	63.56	2.70	0	0	58

Tabla 6. Resultados del proceso de ocultamiento en la imagen 3

Imagen 4: brain_001.dcm 256x256x16							
Método	Capacidad de Ocultamiento (bits)	Capacidad Real (bits)	PSNR (dB)	Tiempo de Ejecución (seg)	*Overflow* (pixeles)	*Underflow* (píxeles)	Control (bits)
Lee	1203	1203	100.80	0.01	0	254	0
Ni	16291	15901	98.31	0.23	0	0	390
Thodi	18114	1074	91.44	55.04	0	0	17040
Lee+Sh	1203	1167	95.00	0.02	0	0	36
P1	2775	2739	94.08	1.45	0	0	36
P2	18085	18049	93.90	1.53	0	0	36
P3	27839	27729	87.65	3.34	0	0	110

Tabla 7. Resultados del proceso de ocultamiento en la imagen 4

Imagen 5: brain_020.dcm 256x256x16							
Método	Capacidad de Ocultamiento (bits)	Capacidad Real (bits)	PSNR (dB)	Tiempo de Ejecución (seg)	*Overflow* (pixeles)	*Underflow* (píxeles)	Control (bits)
Lee	1452	1452	100.80	0.01	0	190	0
Ni	16025	15635	98.44	0.23	0	0	390
Thodi	18578	1042	91.56	56.69	0	0	17536
Lee+Sh	1452	1416	95.02	0.02	0	0	36
P1	3305	3269	94.13	1.41	0	0	36
P2	18526	18490	93.97	1.51	0	0	36
P3	28630	28520	87.71	3.30	0	0	110

Tabla 8. Resultados del proceso de ocultamiento en la imagen 5

El método de Thodi muestra un buen rendimiento en cuanto a capacidad de ocultamiento sin distorsiones visibles sobre la imagen médica. Una gran desventaja es que incrementa notablemente el tiempo de ejecución a medida que las imágenes son de mayor tamaño y esto por el tiempo de procesamiento requerido para encontrar las localidades en donde se ocultarán los datos. Otra desventaja es el gran número de bits que el método necesita guardar dentro de la misma imagen para la exacta recuperación de la imagen original; así los bits de control disminuyen la capacidad real de ocultamiento.

El método de Lee tiene una manera muy sencilla de ocultar y extraer los datos al igual que recuperar la imagen. La desventaja es la poca capacidad de ocultamiento que presenta en comparación con los otros métodos. También se puede notar que todas las imágenes de prueba presentan problemas de desbordamiento, caso que el método original Lee no tiene contemplado. El algoritmo Lee+Sh requiere de pocos bits de control para garantizar la exacta recuperación de la imagen original y logra eliminar los problemas de desbordamiento que muestran las imágenes con el método original. Lee+Sh logra este objetivo con el costo de aumentar ligeramente la distorsión de la imagen.

Con el método P1 se obtienen mejores resultados comparado con el método Lee, pero con P2, donde se ocupan las regiones negras de 2x2, se logra alcanzar y exceder los resultados de los otros métodos sobre las imágenes radiológicas probadas.

Por último con P3, se muestra que es posible incrementar aun más la capacidad real de ocultamiento sin distorsiones visibles a la imagen, también sin problemas de desbordamiento y recuperando exactamente la imagen original.

De esta forma se demuestra que este nuevo método reversible tiene una gran capacidad de ocultamiento de información, superando así las desventajas de los mejores métodos encontrados en la literatura.

5.2 Implementación Hardware

Para la implementación en hardware del algoritmo propuesto (llamado anteriormente P2) se siguió el flujo de diseño mostrado en la figura 21.

El bloque de diseño comprende seleccionar el lenguaje de descripción de hardware, en nuestro caso VHDL, identificar las jerarquías del diseño, realizar simulaciones funcionales, síntesis y optimización del diseño. Posteriormente sigue la implementación, verificando el rendimiento en general del diseño y realizando a su vez modificaciones para optimizarlo. Finalmente se programa el dispositivo a ser utilizado. Una manera de probar el dispositivo es mediante la técnica hardware-en-el-ciclo en donde se utiliza una aplicación en software como interfaz al usuario pero el procesamiento de los datos se realiza directamente en el FPGA. Esta técnica fue utilizada para las pruebas del dispositivo mediante la herramienta de Simulink/Matlab.

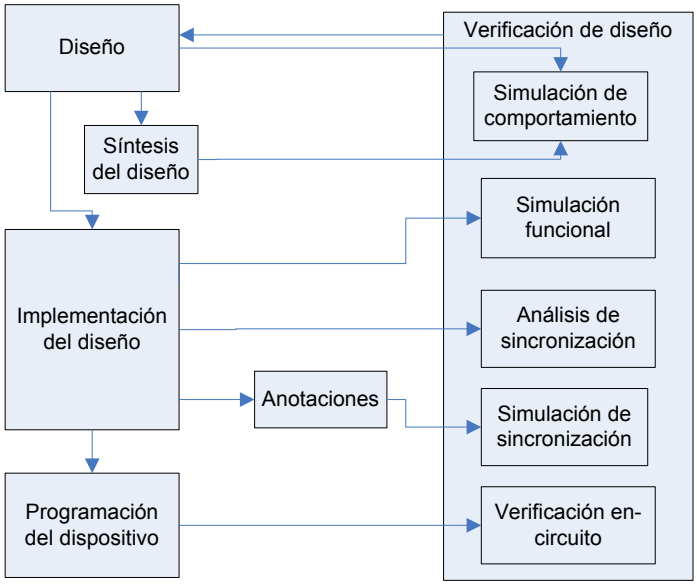


Figura 21. Flujo de diseño [39]

5.2.1 Descripción de la Arquitectura

Para mostrar el diseño de la arquitectura primero se describirá el codificador en general seguido de los detalles arquitecturales. La figura 22 representa el diagrama de bloques de la arquitectura.

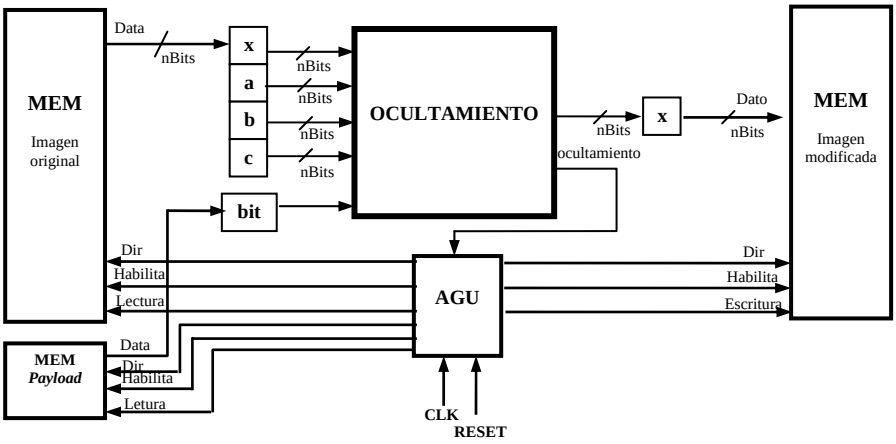


Figura 22. Diagrama de bloques de la arquitectura del método propuesto

5.2.1.1 Módulo *AGU*

El módulo *AGU* genera la dirección de la memoria de donde se obtiene el valor del píxel correspondiente a ser usado en el bloque del proceso de ocultamiento. Se necesita una ventana de 2x2 píxeles formada por el píxel de referencia y su contexto, es decir, su vecino de la derecha, de abajo y de abajo a la derecha, es decir 4 píxeles a la vez en el proceso de ocultamiento. A esta ventana llamaremos localidad actual. También se genera la dirección del siguiente bit a ocultar, esta dirección se actualiza cada vez que la bandera *ocultamiento* lo indique. La bandera *ocultamiento* es igual a '1' si es

posible ocultar un bit en esa localidad con el método propuesto y la bandera es igual a ‘0’ en caso contrario.

Cuando se obtiene completamente la localidad actual el bloque siguiente la procesa y genera el valor modificado del píxel de referencia, y se obtiene también el valor de la bandera *ocultamiento*.

5.2.1.2 Módulo Proceso de Ocultamiento

El diagrama de este módulo se representa en la figura 23. Los datos de entrada a este bloque son la localidad actual, es decir el píxel de referencia y su contexto y el bit a ocultar.

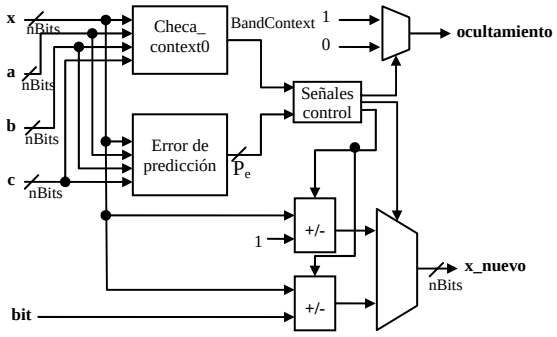


Figura 23. Diagrama de bloques del proceso de ocultamiento

El módulo *Checa_context0* verifica si la localidad actual es una región ‘negra’ o no y genera una bandera *BandContext* que indica si se utiliza esa localidad para insertar o se le hace un corrimiento, véase la tabla 9.

BandContext	**Significado**
00	Es una región “negra” y se usa para ocultar un bit
01	Sólo el contexto es “negro” pero el píxel de referencia no, así que genera un corrimiento a derecha
11	En otro caso

Tabla 9. Valores de la bandera del bloque checa_context0

El módulo de *Error de predicción* evalúa los valores de los píxeles de entrada y obtiene el valor del error de predicción. El módulo de *Señales de control* simplemente es para resumir los comparadores usados para la selección de la operación a realizar, esto es, si el valor del píxel de referencia será modificado con el valor del bit a ser ocultado, o si requiere de algún corrimiento a la derecha o izquierda o simplemente no se modifica, véanse las tablas 10a y 10b. Los datos de salida de este módulo son *x_nuevo* que es el valor modificado del píxel de referencia y *ocultamiento* que indica si se ocultó un bit o no.

Modificación a realizar	**Parámetros usados**
x+1	$p_e \geq 2$ ó *BandContext*="01"
x-1	$p_e \leq -2$
x+bit	$p_e = 1$ ó *BandContext*="00"
x-bit	$p_e = -1$
x	En otro caso

(a)

Valor de *ocultamiento*	**Parámetros usados**
'1'	*BandContext*="00" ó (*BandContext*="11" y (p_e=1 ó p_e=-1))
'0'	En otro caso

(b)

Tabla 10. Valores de los datos de salida según el bloque *señales de control*

El diseño del proceso de extracción y recuperación es similar al procesamiento de ocultamiento, obteniendo el valor del píxel original y el bit oculto en caso de que se haya efectuado un ocultamiento en esa localidad.

5.2.2 Resultados de la Implementación Hardware

Como se mencionó anteriormente, el diseño fue modelado usando herramientas de síntesis y se obtuvieron los resultados mostrados en la tabla 11, ahí se muestran los requerimientos del proceso de ocultamiento.

Las pruebas de la implementación fueron realizadas con Simulink con la técnica Hardware-in-the-loop. Los pasos de las pruebas consistieron en procesar las mismas imágenes que en la implementación en software y posteriormente comparar los resultados obtenidos en cada imagen. En todos los casos se obtuvo la misma imagen modificada y se recuperaron los bits ocultos así como la imagen original.

Periodo de Clk (ns)	18.913
Uso de Celdas (BELS)	467
Slices	151
Slice FFs	81
LUTs	239
IOs	83

Tabla 11. Resumen del reporte de la síntesis del procesamiento de ocultamiento

Posteriormente se compararon los tiempos de ejecución de ambas implementaciones, software (lenguaje C y MATLAB) y hardware. La tabla 12 presenta los tiempos de ejecución de las imágenes presentadas anteriormente. Los tiempos incluyen solamente el procesamiento de ocultamiento.

	Imagen1	**Imagen2**	**Imagen3**	**Imagen4**	**Imagen5**
Tiempo de Ejec. en Matlab (seg)	0.0701	1.8126	1.3222	1.5322	1.5122
Tiempo de Ejecución en C (seg)	0.01731	0.01154	0.11435	0.01100	0.01138
Tiempo de Ejecución en HW (seg)	0.00123	0.00123	0.01983	0.00123	0.00123

Tabla 12. Comparación de los tiempos de ejecución entre SW y HW

Estos resultados fueron obtenidos con la implementación del diseño básico propuesto y usando las imágenes de prueba mostradas. Así, se ve que la implementación en hardware es aproximadamente 10 veces más rápida que la programada en software.

Aún así, es posible optimizar el diseño propuesto para obtener un mejor rendimiento. Con dicho objetivo es posible utilizar una versión paralelizada del diseño en la que se procesen las imágenes por bloques o incluso varias imágenes a la vez, véase figura 24.

Dependiendo de las características requeridas por una aplicación específica y teniendo en cuenta las limitaciones de almacenamiento del hardware se implementaría alguna de estas modificaciones mencionadas. Por un lado, para las imágenes de mayor tamaño (2048x2048 píxeles) se podría realizar un procesamiento por bloques, en donde cada bloque se trataría como una imagen por separado. En caso de que se procesen imágenes pequeñas (256x256, 512x512, etc.) se procesarían varías imágenes a la vez. Para ambos casos el bloque AGU debe ser capaz de direccionar a la imagen para obtener los píxeles a procesar en cada ciclo de reloj.

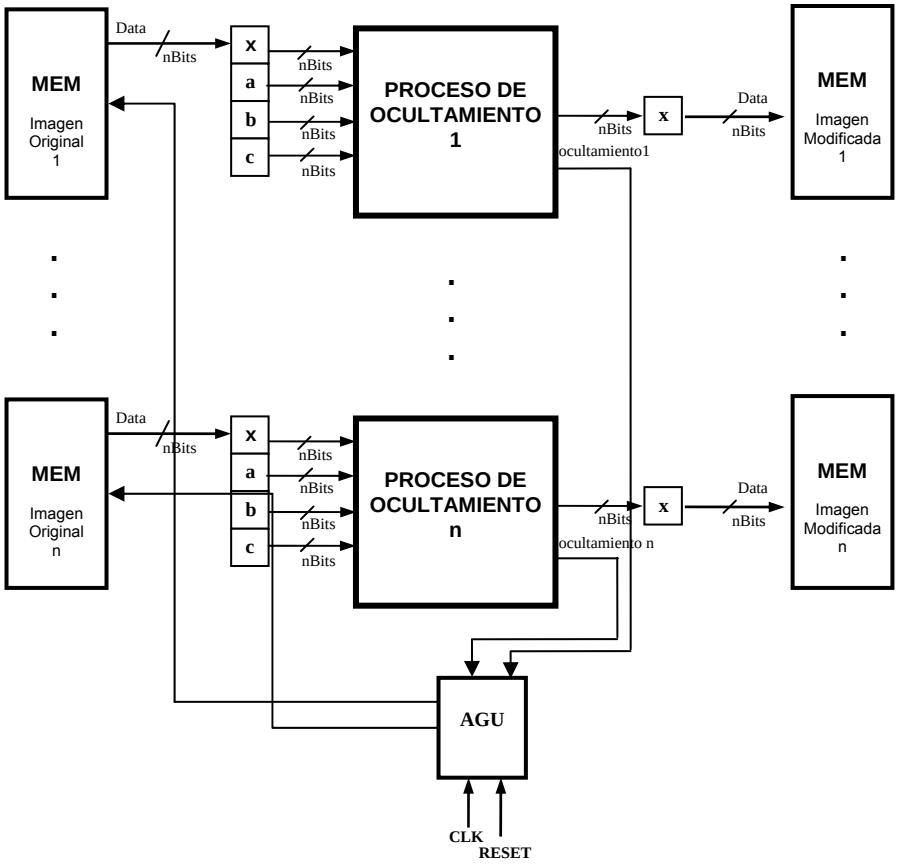


Figura 24. Modificación al diseño mediante replicamiento

Por otro lado el mismo bloque debe tener acceso a la memoria que contenga los datos a ocultar; es decir si se trata de una sola imagen procesada por bloques será necesario un control adicional que indique qué bits serán los siguientes a ocultar para cada bloque, o en cuyo caso direccionar a la memoria que contenga los datos a ocultar para cada imagen.

De esta manera, usando una versión paralelizada del modelo descrito, se tendría un procesamiento más rápido dependiendo del número de imágenes o bloques a procesar.

CAPÍTULO 6
CONCLUSIONES

El ocultamiento de datos reversible es una técnica de seguridad que se ha adoptado ampliamente, principalmente en aplicaciones que involucran imagenología sensible. Muchas de esas aplicaciones se encuentran en el ámbito médico en donde la demanda de sistemas más seguros sigue incrementándose.

En la actualidad se ha propuesto una gran cantidad de métodos de ocultamiento reversible pero no todos son adecuados para todo tipo de imágenes. Por ejemplo, existen métodos que al ocultar información generan ruido del tipo sal y pimienta sobre la imagen y dicha distorsión es visible en imágenes que tienen valores de píxeles homogéneos. Por otro lado, hay métodos que se enfocan en causar una menor distorsión en la imagen, pero generan poca capacidad de ocultamiento. Hay otros métodos donde se tiene gran capacidad de ocultamiento generando poca distorsión a la imagen pero su procesamiento es más costoso y ello incrementa el tiempo de ejecución en imágenes de tamaño grande.

Los métodos existentes tienen ventajas unos sobre otros, pero a su vez cada uno de ellos tiene algunas desventajas. Por ello en este trabajo se

propuso un método capaz de cubrir las principales debilidades de los métodos reversibles de ocultamiento de datos más sobresalientes. Para lograr una mejora se realizó un análisis a los métodos existentes más destacados. En base a sus publicaciones, los que muestran mejores resultados en cuando a capacidad de ocultamiento con menor distorsión en la imagen son, los mostrados en este documento, Thodi, Ni y Lee.

Para poder homogeneizar los resultados se procedió a probar los métodos con el mismo banco de imágenes. En este análisis se tomaron en cuenta las características referentes a la capacidad real de ocultamiento, el tipo de control necesario para la extracción de los datos y la recuperación de la imagen original, y la distorsión de la imagen original generada al ocultar la información. Con esto se lograron identificar las ventajas y desventajas que tiene cada método analizado y de esta manera fue posible proponer un método basado en una sinergia entre las principales características de los tres métodos mencionados explotando así sus ventajas y mejorando esencialmente las desventajas individuales.

Por un lado, como ventajas del método Lee es que tiene un control sencillo para la recuperación, no necesita guardar algún mapa de localización o alguna otra información, el PSNR de la imagen marcada es mayor o igual a 51dB y tiene adecuada capacidad de ocultamiento en imágenes con valores de píxeles homogéneos, sin embargo, este método no propone cómo manejar los posibles problemas de desbordamiento, lo que ocasiona que en algunas imágenes no sea posible recuperar exactamente la imagen original debido a las pérdidas de los valores originales generadas por dicho desbordamiento.

Del método Lee se toma el tipo de control para la recuperación, ya que es sencillo y no requiere de almacenar información adicional. El problema de posibles desbordamientos se logra solucionar mediante un preprocesamiento a la imagen basado en un corrimiento al histograma. Este preprocesamiento se basa en la manera en que el método Ni genera “huecos” para poder

ocultar información mediante corrimientos al histograma de la imagen. Los corrimientos generan un estrechamiento del histograma, lo que evita posibles desbordamientos en los valores de los píxeles después de la inserción de bits de información.

Para el proceso de inserción, se vio que el método Ni no es buen candidato ya que este método ya ha sido explotado y es posible que no se obtenga una mayor capacidad de ocultamiento que la que ya presenta. El método Thodi, por su parte, genera una menor distorsión a la imagen con gran capacidad de ocultamiento, utiliza la técnica de expansión del error de predicción. La principal desventaja de este método (Thodi) es que requiere de bits de control para recuperar la imagen original y por ello debe reservar una región de la misma imagen garantizando que existirá espacio para ocultar los bits de control adicionales; con esto decrece sustancialmente la capacidad real de ocultamiento y al mismo tiempo se incrementa el tiempo de ejecución del algoritmo para imágenes grandes.

Para el algoritmo propuesto se tomó la ventaja de la expansión del error de predicción de Thodi con un umbral fijo similar a la manera en que lo hace Lee. Con estas características el método propuesto cubre las necesidades de la imagenología de tipo sensible tales como gran capacidad de ocultamiento con menor distorsión de la imagen y recuperando la imagen original mediante un control simple.

Los resultados obtenidos de las implementaciones en software de los métodos analizados y el propuesto muestran que el método propuesto alcanzó y excedió el rendimiento de los otros métodos probados con imágenes médicas radiológicas, con una mayor capacidad de ocultamiento real sin distorsiones visibles en la estego imagen, con valores de PSNR altos (arriba de 80 dB) y con una simple solución a los problemas de desbordamiento.

La complejidad del algoritmo es lineal (O(M-1 x N-1), para una imagen de MxN píxeles) así que el método propuesto se puede implementar como

sistema embebido en diversas aplicaciones relacionadas con la imagenología sensible, especialmente las que requieren de sistemas sencillos y pequeños. Las tendencias de sistemas embebidos médicos y de cuidados de la salud requieren cada vez más de algoritmos y arquitecturas especializadas. Con una aproximación en este sentido se exploró una implementación hardware para mostrar la factibilidad en este tipo de sistemas.

De los primeros resultados obtenidos del diseño de la arquitectura hardware muestran que la velocidad de procesamiento se incrementa en un factor de 10. Aun con esto, es posible optimizar el diseño de la arquitectura mostrado en este trabajo, de manera que se pueda procesar una imagen de tamaño grande por bloques o incluso varias imágenes a la vez, ya que la naturaleza del método propuesto no muestra dependencia de datos entre las regiones de píxeles que se procesan.

De esta manera el sistema explorado podría funcionar como un coprocesador para la transmisión en tiempo real de información médica confiable en sistemas móviles.

Con base en estas observaciones y resultados se concluye que el método propuesto satisface exitosamente las necesidades de gran capacidad de ocultamiento de datos garantizando la imperceptibilidad de los mismos y con un control sencillo para la recuperación de la imagen original brindando seguridad en aplicaciones relacionadas con imagenología sensible.

6.1 Trabajo Futuro

El método propuesto se probó especialmente sobre imágenes radiológicas, pero ya que el método se puede aplicar en general para cualquier tipo de imágenes se podría probar con un mayor número y variedad de imágenes sensibles, tales como imágenes satelitales, de reconocimiento

militar y de exploración espacial. Con estas pruebas se presentarían resultados más generales y no dependientes de un solo tipo de imágenes.

Aún cuando el preprocesamiento realizado a la imagen para eliminar los problemas de desbordamiento es eficiente, se podría explorar alguna otra manera de eliminar dichos problemas, de tal forma que no se involucre un procesamiento a la imagen pero sin utilizar un control complejo.

Así mismo, se debe efectuar un análisis a las características que tienen los otros tipos de imagenología sensible, diferentes a las imágenes radiológicas, de tal manera que se pueda generalizar la mejora efectuada al método propuesto, en la que se insertan datos en regiones “negras”, aumentando así la capacidad de ocultamiento.

Por otro lado, se propone la optimización de la arquitectura presentada en este documento, utilizando varías técnicas para mejorar el rendimiento de la arquitectura presentada; un ejemplo sería utilizar una versión paralelizada del diseño básico, tal como se muestra al final del capítulo 5.

Apéndices

A- Implementaciones Software

Este apéndice comprende todos los códigos de programación para las implementaciones software, tanto de los métodos tomados como base como del método propuesto. Estas implementaciones fueron utilizadas para obtener resultados de las mismas imágenes de prueba y hacer así una comparación entre ellos.

A1- Códigos del Método Lee

Escanea la imagen en un orden, por cada par de pixeles obtiene la diferencia de los valores, si Diff>=2 entonces hace un corrimiento a la derecha, si Diff<=-2 hace un corrimiento a la izquierda, pero si la diferencia es 1 ó -1, utiliza este par para ocultar un bit dela marca. Nota: no maneja problemas de over/underflow al hacer los corrimientos

```
function [cont0,cont255,Image,psnr,EmbedCap, Tiempo]= diferenciasLee(Image, data, Bits)
[row,column]=size(Image);
d=0;
L=2^Bits-1;
for i=1:row
   for j=1:floor(column/2)
            impar=double(Image(i,2*j-1));
            par=double(Image(i,2*j));
            Diff=impar - par;
            if Diff>=2
                     Image(i,2*j-1)= Image(i,2*j-1) +1;   % impar= impar + 1
            elseif Diff<=-2
                     Image(i,2*j-1)= Image(i,2*j-1) -1;   % impar= impar - 1
            elseif Diff==-1 && d<= tam_d
                     d=d+1;  %avanza contador de data
            if data(d)==1
                     Image(i,2*j-1)= Image(i,2*j-1) -1;
            end
            elseif Diff==1 && d<= tam_d
                     d=d+1;  %avanza contador de data
            if data(d)==1
                     Image(i,2*j-1)= Image(i,2*j-1) +1;
            end
            end
   end
end
```

A2- Códigos del Método Lee más Mejora Propuesta

A2.1 Proceso de Inserción

Rastrea la imagen, genera el histograma y encuentra los valores cero o mínimos del histograma y ocupa ese espacio para hacer un corrimiento

```
[LMin,LMax,PixelUno] = ConHistogram (Image,Bits);

for i=1:row
  for j=1:floor(column/2)
      if Image(i,j)==PixelUno
          %guarda el pixel que borrara
          PixelSave=[i j];
      end
      %realiza el corrimiento
      if Image(i,2*j-1)<LMin
          Image(i,2*j-1)=Image(i,2*j-1)+1;
      elseif Image(i,2*j-1)>LMax
          Image(i,2*j-1)=Image(i,2*j-1)-1;
      end
      if Image(i,2*j)<LMin
          Image(i,2*j)=Image(i,2*j)+1;
      elseif Image(i,2*j)>LMax
          Image(i,2*j)=Image(i,2*j)-1;
      end
          %Sigue el bloque del apéndice A1
 end
end
```

A2.2 Proceso de recuperación (Lee + mejora)

```
function [Image,psnr] = diferenciasLee_dec (Image,Embed)
 [row,column]=size(Image);
%se deben obtener los valores de LMin y LMax del inicio de la marcarecuperada
for i=1:row
   for j=1:floor(column/2)
          impar=double(Image(i,2*j-1));
          par=double(Image(i,2*j));
          Diff=impar - par;
          if Diff==-1 || Diff ==1
                  d=d+1;
                  Embed(d)=0;
          elseif Diff==-2
                  d=d+1;
                  Embed(d)=1;
                  Image(i,2*j-1)=Image(i,2*j-1)+1;
          elseif Diff==2
                  d=d+1;
                  Embed(d)=1;
                  Image(i,2*j-1)=Image(i,2*j-1)-1;
          end
          if Diff>2
                  Image(i,2*j-1)= Image(i,2*j-1) -1;   % impar= impar + 1
          elseif Diff<-2
                  Image(i,2*j-1)= Image(i,2*j-1) +1;   % impar= impar - 1
          end
          %recupera el histograma original
          if Image(i,2*j-1)<=LMin
                  Image(i,2*j-1)=Image(i,2*j-1)-1;
```

```
                elseif Image(i,2*j-1)>=LMax
                        Image(i,2*j-1)=Image(i,2*j-1)+1;
                end
                if Image(i,2*j)<=LMin
                        Image(i,2*j)=Image(i,2*j)-1;
                elseif Image(i,2*j)>=LMax
                        Image(i,2*j)=Image(i,2*j)+1;
                end
                end
        end
        EmbedCap=d
```

A3- Códigos del Método Ni más Mejora

A3.1 Proceso de inserción

Image= es el bloque de la imagen original, data= archivo de los datos a insertar; O= tamaño total del archivo a insertar, d=contador de los datos insertados, I= el bloque de la imagen con los datos insertados, Embed es la cantidad de datos insertados, Z y P son los arreglos que contienen a los pares de zeros y peaks

```
function [I,d,Z_1,P_1]=Ni_kp (Image,data,O,d,Bits,BitsStorage)
% Genera el histograma
n=histograma(Image,Bits);
%3. Inicializa k
k=1;
fail=0;
[row,column]=size(Image);
%4. Encuentra min y máx en el histograma
enough=0;   %bandera de paro del ciclo while
pay_max=0;
payload=0;
countk=0;
MaximoP=0;
while ~enough && ~fail
        if k==1
                prev_min=0; %1
        else
                prev_min=minimum;
        end %if k==1
        minimum=row*column;
        for i=1:2^Bits
                if (n(i)<minimum && n(i)>=prev_min )
                        if k==1 % tratando de asegurar que los Z's no estén juntos
                                minimum = n(i);
                                Z(k)=i;
                        else
                                Bverif = verifica (i,Z,k-1);
                                B1verif=verifica(i,P,numel(P));
                                if Bverif && B1verif
                                        minimum = n(i);
                                        Z(k)=i;
                                end %if Bverif
                        end % if k==1
```

```
                end % if
        end %for i
        %% ordena Z
        Z=sort(Z);
        if k==1
                prev_max=row*column;
        else
                prev_max=maximum;
        end
        maximum=0;
        for i=1:256
                if (n(i)>maximum && n(i)<= prev_max)
                        if k==1
                                maximum = n(i);
                                P(k)=i;
                                MaximoP=i;
                        else
                                Bverif=verifica(i,P,k-1);
                                B1verif=verifica(i,Z,numel(Z));
                                if Bverif && B1verif
                                        maximum = n(i);
                                        P(k)=i;
                                end
                        end % if
                end % if
        end %for
    P=sort(P);
   if numel(Z)<k  || numel(P)<k
        enough=1;
   end
   %5. hay suficiente capacidad??
   pay_max=pay_max+maximum;
   payload= pay_max - O;
   if payload >=0 || k==10
        enough=1;
   end %payload>0
   k=k+1;
end % while ~enough
%Esta función verifica que los pares de Z y P no se traslapen, en caso de serlo elimina los pares traslapados
[P,Z]=traslapados(P,Z, MaximoP);
k_pair=min(numel(Z),numel(P));
% cuenta nuevamente cual es el pay-max y recorre los valores para el histograma uno a la izquierda
pay_max=0;
for m=1:k_pair
        pay_max=pay_max+ n(P(m));
        Z(m)=Z(m)-1;
        P(m)=P(m)-1;
end
%6. recorre el histograma
k=1;
d_aux=1;
for k=1:k_pair
        if P(k) > Z(k)
                for i=1:row
                        for j=1:column
                        %recorre el histograma a la izquierda
                                if Image (i,j)>=Z(k)+1 && Image (i,j)<=P(k)
                                        Image (i,j)= Image (i,j)-1;
                                end % if image
                        end %for j
                end %for i
        else %if P<Z
                for i=1:row
                        for j=1:column
                        %recorre el histograma a la derecha
                                if Image (i,j)>=P(k)+1 && Image (i,j)<=Z(k)-1
                                        Image (i,j)= Image (i,j)+1;
```

```
                                        end % if image
                                end %for j
                        end %for i
                end % if
        %5. Inserta el bit en el hueco formado
        for i=1:row
                for j=1:column
                        if d_aux<=pay_max   % el máximo numero de bits insertados
                                if P(k) > Z(k) && Image(i,j)==P(k)-1
                                %entonces el valor del pixel se incrementa en uno si el bit del dato es '1'; si es cero,
                                %entonces no cambia
                                        if data(d)==1
                                                Image(i,j)=Image(i,j)+1;
                                                d=d+1;
                                                d_aux=d_aux+1;
                                        else
                                        d=d+1;
                                        d_aux=d_aux+1;
                                end %if data(d)
                        end % if P>Z
                        if P(k) < Z(k) && Image (i,j)== P(k)
                                if data(d)==1
                                        Image(i,j)=Image(i,j)+1;
                                        d=d+1;
                                        d_aux=d_aux+1;
                                else
                                        d=d+1;
                                        d_aux=d_aux+1;
                                end %if data(d)
                        end % if P<Z
                        end % d<=O
                end %for j
        end %for i
end % for k
```

A3.2 Proceso de recuperación (Ni)

Algoritmo: 1. Lee la imagen 2. Encuentra los Zeros y Picos y se obtiene el valor del dato insertado 3. Recorre el histograma 4. Obtiene la imagen original

```
function [I,data_recover,d]=Ni_kp_reverse (Image,Z,P,max_d,d)
k_pair= P(1); % numero de pares totales
[row,column]=size(Image);
d1=1; % se lleva un contador auxiliar para regresar los datos en orden
for k=2:k_pair+1
        if P(k)>Z(k)
                for i=1:row
                        for j=1:column
                                if d<=max_d
                                        if Image(i,j)==P(k)
                                        %dato es '1' y no cambia el valor del pixel
                                                data_recover(d1,1)=1;
                                                d1=d1+1;
                                                d=d+1;
                                        end % if Image=P

                                        if Image(i,j)==P(k)-1
                                                data_recover(d1,1)=0;
                                                d1=d1+1;
                                                d=d+1;
                                                Image(i,j)= Image (i,j)+1;
```

```
                                        end  % if Image=P-1
                                        if Image(i,j)>=Z(k) && Image(i,j)<=P(k)-2
                                                Image(i,j)=Image(i,j)+1;
                                        end % If Image(i,j) in [Z,P-2]
                                end % if d<=tam
                        end %for j
                end % for i
                else % Z>P
                        if P(k)<Z(k)
                                % entonces mismo procedimiento que si P>Z con corrimientos a la izquierda
end % for
```

A4 Códigos del Método Thodi

A4.1 Proceso de inserción

1.- Se realiza un preprocesamiento para separar las regiones R y S

2.- Se genera el bitstream B= P U EOP U R_LSB

3.- Proceso de inserción

a) busca el error de predicción

b) selecciona a que conjunto pertenece: E, N ó U

c) inserta los bits en los píxeles de E

```
%1.- Se realiza un preprocesamiento para separar R y S
for i=1:row-1
   for j=1:column-1
      x=double(Image(i,j));
      a=Image(i+1,j);
      b=Image(i,j+1);
      c=Image(i+1,j+1);
      [x_pred,Pe] = predicted (x,a,b,c);
      x_new=x+Pe;
      %si pertenece a E
      if (abs(Pe)<=T) && (x_new>=0) && (x_new <= L-2)
            emCapacity= emCapacity+1;
      %si pertenece a Ue
       if (0< x_new  && x_new <= T-1) || (L-1-T<= x_new && x_new < L-1) || (0< x_new+1  && x_new+1 <= T-1) ||
                        (L-1-T<= x_new+1 && x_new+1 < L-1)
            Rc=Rc+1;
       end
      %Si pertenece a N (fuera del umbral)
      elseif abs(Pe)>T
         %si pertenece a Ue
         if (0< x_new  && x_new <= T-1) || (L-1-T<= x_new && x_new < L-1) || (0< x_new+1  && x_new+1 <= T-1) ||
                        (L-1-T<= x_new+1 && x_new+1 < L-1)
            Rc=Rc+1;
         end
      %si pertenece a U
      else
            Rc=Rc+1;
      end
   end
end

%Reserva la region R y establece desde donde empieza S, es decir la region
%S empieza en Image(Fila, Columna) y termina hasta Image(row, column)
```

```
Reserve=[];
k=1;
for i=1:row
   for j=1:column
      if k<=Rc
         Reserve=[Reserve i j];
         k=k+1;
         %checa la capacidad
          if (i<=row-1 && j<= column-1)
            x=double(Image(i,j));
            a=Image(i+1,j);
            b=Image(i,j+1);
            c=Image(i+1,j+1);
            [x_pred,Pe] = predicted (x,a,b,c);
            x_new=x+Pe;
            %si pertenece a E
            if (abs(Pe)<=T) && (x_new>=0) && (x_new <= L-2)
               emCapacity= emCapacity-1;
            end
          end
      end
   end
end
%2.- Se genera el bitstream B= P U EOP U R_LSB
EOP=[1 0 1 0 0 0 1 0 1 1 1 1 0 0 1 0 0 0 0 0 1 0 1 0];
%a) colecta el LSB de R
k=0;
R_LSB=[];
for i=1:row
  for j=1:column
      if k< Rc
         k=k+1;
         if (mod(Image(i,j),2)==0)
            R_LSB(k)=0;
         else
            R_LSB(k)=1;
         end
      end
  end
end
%b) capacidad sobrante para el payload
Capacity= emCapacity - Rc - 24;
P=data(1:Capacity);
P=P.';
B=[P EOP R_LSB];
Cuenta=0;
%3.- Proceso de insercion sobre la region S, que va de Image(row_ini:row-1,column_ini:column-1)
%   a) busca el error de predicción
for i=1:row-1
   for j=1:column-1
     enR=isR(i,j,Reserve);
     if enR==0     % Si enR=0 ent esta en S y es posible insertar
       x=double(Image(i,j));
       a=Image(i+1,j);
       b=Image(i,j+1);
       c=Image(i+1,j+1);
       [x_pred,Pe] = predicted (x,a,b,c);
       x_new=x+Pe;
       %   b) selecciona a que conjunto pertenece: E, N ó U
       %si pertenece a E (expandible)
       if (abs(Pe)<=T) && (x_new>=0) && (x_new <= L-2)
          if numel(B)>0
             x_new=x+Pe+B(1);    %inserta el bit x'=x+Pe+i
             B=B(2:numel(B));    % se tira del Bitstream
             Image(i,j)= x_new;  %modifica el valor del pixel
             %si pertenece a Ue
```

```
            if (0< x_new  && x_new <= T-1) || (L-1-T<= x_new && x_new < L-1)
               B=[1 B]  %inserta bit bandera al inicio de B
               Cuenta=Cuenta+1;
            end
         end
      %Si pertenece a N hace un corrimiento del histograma
      elseif abs(Pe)>T
         %corrimiento
         if Pe>=0
            x_new= x+T;
         else
            x_new=x-T+1;
         end
         %si no hay under/overflow cambia el valor del pixel
         if (0<=x_new && x_new<=L-1)
            Image(i,j)=x_new;
         end
         %si pertenece a Ue
         if (0< x_new  && x_new <= T-1) || (L-1-T<= x_new && x_new < L-1)
            B=[1 B];    %inserta bit bandera al inicio de B
             Cuenta=Cuenta+1;
         end
      else % No cambia
          B=[0 B];
          Cuenta=Cuenta+1;
      end
    end
  end
end
%Si falta insertar datos del bitstream, se insertan en el LSB  de los píxeles al inicio de
%la imagen
mascara0=L-2;
for i=1:row
   for j=1:column
      enR= isR(i,j,Reserve);
      if (numel(B)>0 && enR==1) % si hay elementos en B e i,j
                        %inserta el bit 1 de B en el LSB
                        x_new=2*floor(Image(i,j)/2)+B(1);
         if numel(B)>1 % Tira el bit insertado de B
            B=B(2:numel(B));
         else
            B=[];
         end
         Image(i,j)=x_new;   %modifica el valor del pixel
      else
         break;
         break;
      end
   end
end
```

A5 Códigos del Método Propuesto P1

% este programa sigue la idea de predecir un valor dentro del contexto de un píxel para que la diferencia sea menor
%entre el píxel de referencia y el predicho, así también sigue el principio de encontrar huecos para hacer
%corrimientos al histograma y así evitar under/overflow

```
function [Image]=diferenciasLeeNiJezTras(Image, data, Bits)
[row,column]=size(Image);
d=0;
L=2^Bits-1;
%encuentra los huecos en el histograma para preprocesamiento
[LMin,LMax,Sirve,PixelUno] = ConHistogram (Image,Bits);
%Realiza el corrimiento
```

```
for i=1:row
   for j=1:column
      if Image(i,j)<LMin
         Image(i,j)=Image(i,j)+1;
      elseif Image(i,j)>LMax
         Image(i,j)=Image(i,j)-1;
      end
   end
end
% inicia proceso de ocultamiento
for i=1:row-1
    for j=1:column-1
      x=double(Image(i,j));
      a=Image(i+1,j);
      b=Image(i,j+1);
      c=Image(i+1,j+1);
      [x_pred,Diff] = predicted (x,a,b,c);
      if Diff>=2
         x_new= x +1;
      elseif Diff<=-2
         x_new= x -1;
      elseif Diff==-1
            d=d+1;  %avanza contador de data
            if data(d)==1
               x_new=x-1;
            else
               x_new=x;
            end
      elseif Diff==1
          d=d+1;  %avanza contador de data
          if data(d)==1
            x_new=x+1;
          else
             x_new=x;
          end
      else
         x_new=x;
      end
    Image(i,j)=x_new;
   end
end
```

A6 Códigos del Método Propuesto P2

```
function [Image]=diferenciasLeeNiJezTrasCero(Image, data, Bits);
 [row,column]=size(Image);
d=0;
L=2^Bits-1;
%Realiza el preprocesamiento como en A5 y continua
for i=1:row-1
    for j=1:column-1
      x=double(Image(i,j));
      a=Image(i+1,j);
      b=Image(i,j+1);
      c=Image(i+1,j+1);
      if a==1 && b==1 && c==1 %valores negros mas corrimiento
         if x==1
            d=d+1;  %avanza contador de data
            if data(d)==1
               x_new=x+1;
            else
               x_new=x;
            end
         else
            x_new=x+1;
```

```
        end
      else
%continua el algoritmo como A5
```

A7 Código del Método Propuesto P2 Programado en lenguaje C

```
int i,j,a,b,c,x,d,x_new,x_pred,Diff;
LARGE_INTEGER t_ini, t_fin;  double secs;
QueryPerformanceCounter(&t_ini);
 d=1;
 for(i=0;i<256;i++) {
  for(j=0;j<256;j++)
   {x=ima[i][j];
    a=ima[i+1][j];
    b=ima[i][j+1];
    c=ima[i+1][j+1];
     if ((a==1)&&(b==1)&&(c==1))     {
      if( x==1)     {
        if (d==1)
          x_new=x+1;
         else
          x_new=x;
       }
      else      {
        x_new=x+1;
       }
     }
    else     {
      Diff=pre(x,a,b,c,x_pred);
      if (Diff>=2)     {
       x_new= x +1;
      }
      if (Diff<=-2)        {
        x_new= x -1;
       }
      if (Diff==-1) {
        if (d==1)
         x_new=x-1;
        else
         x_new=x;
       }
      if (Diff==1)        {
       if (d==1)
         x_new=x+1;
        else
         x_new=x;
```

```
      }
      if(Diff==0)      {
         x_new=x;
      }
    }     ima[i][j]=x_new;    }
  }

  QueryPerformanceCounter(&t_fin);
  secs = performancecounter_diff(&t_fin, &t_ini);
  fin->Text= secs;
```

B-Implementaciones Hardware

B1- Códigos VDHL de la Arquitectura

Bloque de ocultamiento de datos:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Hiding_Process is
            generic (nBits:natural:=16);
            Port ( x,a,b,c : in  STD_LOGIC_VECTOR (nBits-1 downto 0);            -- the pixel x and its context (a,b,c)
            clk, hiding_bit : in  STD_LOGIC;
            x_new : out  STD_LOGIC_VECTOR (nBits-1 downto 0);
            Embed: out STD_LOGIC                    ); -- Embed='1' if there is a embedded bit, embed='0' otherwise
end Hiding_Process;

architecture Behavioral of Hiding_Process is
-- if x and its context=0 then hiding_flag is "00"(Embed); if x/=0 but its context is 0 then hiding_flag is "01" (shift x)
-- otherwise hiding_flag is "11" (do nothing)
component check_Context0 is
             generic (n: natural:=8);
             Port ( x,a,b,c : in  STD_LOGIC_VECTOR (n-1 downto 0);
             hiding_flag : out  STD_LOGIC_VECTOR (1 downto 0));
end component;

-- this obtain the predicted error based on the predicted value and x
component PredictionError is
            Generic (k: natural:=8);
            Port ( x,a,b,c : in  STD_LOGIC_VECTOR (k-1 downto 0);
            PredError_out : out STD_LOGIC_VECTOR (31 downto 0) );
end component;

signal SHiding_flag: STD_LOGIC_VECTOR (1 downto 0);
signal PredError: STD_LOGIC_VECTOR (31 downto 0);
signal x_reg,a_reg,b_reg,c_reg, x_new_reg : STD_LOGIC_VECTOR (nBits-1 downto 0);
signal Embed_reg: STD_LOGIC;

begin
reg_in: process (clk)            -- genera registros en la entrada y salida para disminuir el tiempo
            begin
                        if clk'event and clk='1' then
                                    x_reg<= x;
```

```
                                a_reg<= a;
                                b_reg<= b;
                                c_reg<= c;
                                x_new <= x_new_reg;
                                Embed <= Embed_reg;
                        end if;
        end process;

        CheckContextZero:     check_Context0 generic map (n=>nBits)
                                port map (x_reg,a_reg,b_reg,c_reg, SHiding_flag);
        ObtainPredictionError: PredictionError generic map (k=>nBits)
                                port map (x_reg,a_reg,b_reg,c_reg,PredError);

        x_new_reg<= x_reg+1 when (signed(PredError)>=2 or SHiding_flag="01") else
                                 x_reg-1 when signed(PredError)<=-2 else
                                 x_reg+hiding_bit when (signed(PredError)=1 or SHiding_flag="00") else
                                 x_reg-hiding_bit when signed(PredError)=-1 else
                                 x_reg;

        Embed_reg<= '1'     when ((SHiding_flag="11" and (signed(PredError)=1
                                or signed(PredError)=-1))
                                or SHiding_flag="00")
                                else        '0';
end Behavioral;
```

Componente *check_Context0:*

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity check_Context0 is
        generic (n: natural:=8);
        Port ( x,a,b,c : in  STD_LOGIC_VECTOR (n-1 downto 0);
        hiding_flag : out  STD_LOGIC_VECTOR (1 downto 0));
end check_Context0;

architecture Behavioral of check_Context0 is
begin
hiding_flag<= "00" when (a=1 and b=1 and c=1 and x=1)  else
              "01" when (a=1 and b=1 and c=1 and x/=1) else
               "11" ;
end Behavioral;
```

Componente *PredictionError*

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity PredictionError is
        Generic (k: natural:=8 );
        Port ( x,a,b,c : in  STD_LOGIC_VECTOR (k-1 downto 0);
        PredError_out : out STD_LOGIC_VECTOR (31 downto 0)  );
end PredictionError;

architecture Behavioral of PredictionError is

-- calculate the minor or equal value between (a,b)
component MinEq is
        generic( n:natural:=8);
        Port ( a,b : in  STD_LOGIC_VECTOR (n-1 downto 0);
```

```
            min_out : out  STD_LOGIC_VECTOR (n-1 downto 0)  );
end component;

-- calculate the greater or equal value between (a,b)
component MaxEq is
            generic (n:natural:=8);
            Port ( a,b : in  STD_LOGIC_VECTOR (n-1 downto 0);
            max_out : out  STD_LOGIC_VECTOR (n-1 downto 0));
end component;

-- calculate the value adds= a+b-c
component AddSub is
            generic (n:natural:=8);
            Port ( a,b,c : in  STD_LOGIC_VECTOR (n-1 downto 0);
            adds : out  STD_LOGIC_VECTOR (n-1 downto 0));
end component;

-- multiplexor
component mux3to1 is
             generic (n:natural:=8);
            Port ( sel : in  STD_LOGIC_VECTOR (1 downto 0);
                        in1, in2, in3: in  STD_LOGIC_VECTOR (n-1 downto 0);
                        mux_out : out  STD_LOGIC_VECTOR (n-1 downto 0));
            end component;

signal min_ab, max_ab, adds_abc, x_Predicted: std_logic_vector (k-1 downto 0);
signal sel: std_logic_vector (1 downto 0);
signal x1: std_logic_vector (31 downto 0):=(others=>'0');

begin
            Minimus:  MinEq generic map (n=>k)    port map (a,b,min_ab);
            Maximus:  MaxEq generic map (n=>k)   port map (a,b,max_ab);
            Addition: AddSub generic map (n=>k)    port map (a,b,c,adds_abc);
            -- mux3to1 control, for the Medium Error Predictor:
            -- choose min(a,b) if c<=max(a,b); max(a,b) if c>= min(a,b) otherwise a+b-c
            sel<= "00" when c<=min_ab else
                    "01" when c>=max_ab else
                    "11";
            Selection: mux3to1 generic map (n=>k)    port map (sel, max_ab, min_ab, adds_abc, x_Predicted);

            -- change x (8,12 or 16 bits) to 32 bits, for the subtract
            x1<= ("0000000000000000" & x) when k=16 else
                        ("00000000000000000000" & x) when k=12 else
                        ("00000000000000000000000000" & x);

            PredError_out<= x1- x_Predicted;
end Behavioral;
```

Componente *minimus*

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity MinEq is
            generic( n:natural:=8);
            Port ( a,b : in  STD_LOGIC_VECTOR (n-1 downto 0);
            min_out : out  STD_LOGIC_VECTOR (n-1 downto 0)  );
end MinEq;

architecture Behavioral of MinEq is
begin
            min_out<= a when a<=b else b;
end Behavioral;
```

Componente *maximus:*

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity MaxEq is
        generic (n:natural:=8);
        Port ( a,b : in  STD_LOGIC_VECTOR (n-1 downto 0);
        max_out : out  STD_LOGIC_VECTOR (n-1 downto 0));
end MaxEq;

architecture Behavioral of MaxEq is
begin
        max_out<= a when a>=b else b;
end Behavioral;
```

Componente *adition:*

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity AddSub is
        generic (n:natural:=8);
        Port ( a,b,c : in  STD_LOGIC_VECTOR (n-1 downto 0);
        adds : out  STD_LOGIC_VECTOR (n-1 downto 0));
end AddSub;

architecture Behavioral of AddSub is
begin
        adds<=a+b-c;
end Behavioral;
```

Componente *mux3to1:*

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity mux3to1 is
         generic (n:natural:=8);
        Port ( sel : in  STD_LOGIC_VECTOR (1 downto 0);
      in1, in2, in3: in  STD_LOGIC_VECTOR (n-1 downto 0);
      mux_out : out  STD_LOGIC_VECTOR (n-1 downto 0));
end mux3to1;

architecture Behavioral of mux3to1 is
begin
        with sel select
        mux_out<= in1 when "00",
                  in2 when "01",
                  in3 when others;
end Behavioral;
```

B2- Códigos de Definiciones para Pruebas Hardware-en-el-Ciclo

```
function Hiding_Process_config(this_block)

  this_block.setTopLevelLanguage('VHDL');
  this_block.setEntityName('Hiding_Process');
  this_block.tagAsCombinational;
  this_block.addSimulinkInport('x');
  this_block.addSimulinkInport('a');
  this_block.addSimulinkInport('b');
  this_block.addSimulinkInport('c');
  this_block.addSimulinkInport('hiding_bit');
  this_block.addSimulinkInport('clk');
  this_block.addSimulinkOutport('x_new');
  this_block.addSimulinkOutport('Embed');
  Embed_port = this_block.port('Embed');
  Embed_port.setType('UFix_1_0');
  Embed_port.useHDLVector(false);
  % ------------------------------
  if (this_block.inputTypesKnown)
    this_block.addGeneric('nBits', this_block.port('x').width);
    this_block.port('a').setWidth(this_block.port('x').width);
    this_block.port('b').setWidth(this_block.port('x').width);
    this_block.port('c').setWidth(this_block.port('x').width);

    if (this_block.port('hiding_bit').width ~= 1);
      this_block.setError('Input data type for port "hiding_bit" must have width=1.');
    end

    this_block.port('hiding_bit').useHDLVector(false);

    if (this_block.port('clk').width ~= 1);
      this_block.setError('Input data type for port "clk" must have width=1.');
    end

    this_block.port('clk').useHDLVector(false);
    this_block.port('x_new').setWidth(this_block.port('x').width);
  end  % if(inputTypesKnown)
  % ------------------------------

   if (this_block.inputRatesKnown)
     inputRates = this_block.inputRates;
     uniqueInputRates = unique(inputRates);
     outputRate = uniqueInputRates(1);
     for i = 2:length(uniqueInputRates)
       if (uniqueInputRates(i) ~= Inf)
         outputRate = gcd(outputRate,uniqueInputRates(i));
       end
     end  % for(i)
     for i = 1:this_block.numSimulinkOutports
       this_block.outport(i).setRate(outputRate);
     end  % for(i)
   end  % if(inputRatesKnown)
  % ------------------------------

uniqueInputRates = unique(this_block.getInputRates);

  this_block.addGeneric('nBits','natural','16');

  this_block.addFile('AddSub.vhd');
  this_block.addFile('MaxEq.vhd');
  this_block.addFile('MinEq.vhd');
  this_block.addFile('PredictionError.vhd');
  this_block.addFile('check_Context0.vhd');
  this_block.addFile('Hiding_Process.vhd');

return;
```

Lista de Figuras

Lista de Tablas

Referencias

[1] Bender W. et al., "*Application for Data Hiding*". IBM Systems Journal, Vol. 39, Nos. 3&4, 2000, pp 547-568.

[2] Imágenes:
Exploración espacial:
http://grenouille-bouillie.blogspot.com/2007/10/why-we-need-to-go-into-deep-space.html;
Reconocimiento militar:

http://www.military.com/Content/MoreContent1/0,,GH_Iraq_Satellite,00.html
Imagen médica:
S. Barré: Medical Image Samples: *http://barre.nom.fr/medical/samples/*

[3] W. K. Chen, "*The VLSI Handbook"*, CRC Press, August 2000.

[4] A. Menezes, P. van Oorschot and S. Vanstone, "*Handbook of Applied Cryptography*". CRC Press Inc. 1997; pp 3-5

[5] P. Wayner, "Disappearing Cryptography " AP Professional, Chestnut Hill, Mass., 1996.

[6] S. Katzenbeisser, Fabien A.P. Petitcolas, "*Information Hiding Techniques for Steganography and Digital Watermarking*", Artech House, Inc. 2000.

[7] I. Cox, M. L. Miller, "*The First 50 years of Electronic Watermarking*". Journal of Applied Signal Processing, 2, 2002; pp126-132.

[8] Fabien A.P. Petitcolas, R. J. Anderson, M. G.Kuhn, "*Information Hiding- A Survey*"; Proceedings of the IEEE, special issue on protection of multimedia content, 87(7), Julio 1999. pp1062-1078

[9] http://www.petitcolas.net/fabien/steganography/mp3stego/

[10] http://www.freewr.com/freeware.php?download=jphide-and-jpseek&lid=258

[11] http://www.mirrors.wiretapped.net/security/steganography/blindside/

[12] http://www.darkside.com.au/gifshuffle/

[13] http://wbstego.wbailer.com/

[14] http://compression.ru/video/stego_video/index_en.html

[15] 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.

[16] 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.

[17] G. Coatrieux, L. Lecornu, B. Sankur, and C. Roux. **"**A Review of Image Watermarking Applications in Healthcare" Conference Procedure of the IEEE Engineering in Medicine and Biology Society 2006 ;1 :4691-4694

[18] Honsinger, C.W., Jones, P., Rabbani, M., and Stoffel, J. C.: "*Lossless recovery of an original image containing embedded data"*. U.S patent application, Docket No 77 102/E-D, 1999.

[19] Fridrich, J., Goljan, M., and Du, R.: "*Invertible authentication*". Proc. of SPIE 2001, Security and Watermarking of Multimedia Contents III,. Editor(s): Wong, P.W.,Delp, E.J., Vol. 4314, pp. 197-208.

[20] Fridrich, J., Goljan, M., and Du, R.: "*Distortion-free Data Embedding for Images*". Proc. 4th Information Hiding Workshop, Pittsburgh, Pennsylvania, Apr. 25-27, 2001.

[21] Tian, J.: "*Wavelet-based reversible watermarking for authentication*". Proc. of SPIE 2002, Security and Watermarking of Multimedia Contents III. Editor(s): Wong, P.W., Delp, E.J., Vol. 4675, pp. 679-690.

[22] Lee S-K, Suh Y-H, Ho Y-S*,* **"**R*eversible Image Authentication Based On Watermarking"*, IEEE International Conference on Multimedia & Expo (ICME) 2006, Canada, pp 1321-1324.

[23] Thodi D.M., Rodriguez J. J., "*Prediction-error-based reversible watermarking*" *Proc. IEEE Conf. Image Processing*, Oct. 2004, pp. 1549–1552.

[24] Ni Z., Shi Y., Ansari N., and Su W., "*Reversible data hiding*" Proc. ISCAS 2003, vol. 2, pp. 912–915.

[25] 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.

[26] M. Weinberger, G. Seroussi, and G. Sapiro, "*LOCOI: A Low Complexity, Context-Based, Lossless Image Compression Algorithm*" in Proc. of die IEEE Data Compression Conf 1996, pp. 140-149.

[27] Thodi DM, Rodriguez J. Expansion Embedding Techniques for Reversible Watermarking. IEEE Transactions on Image Processing. Vol 16, No 3, March 2007

[28] C.-T. Li, " Reversible Watermarking Scheme with Image-independent Embedding Capacity," *IEEE Proceedings - Vision, Image, and Signal Processing*, vol. 152, no. 6, pp. 779 - 786, 2005.

[29] Alattar, A.M, "Reversible watermark using the difference expansion of a generalized integer transform". IEEE Transactions on Image Processing, Aug. 2004, Vol. 13, pp. 1147- 1156.

[30] Coltuc, D. Chassery, J.-M. "High Capacity Reversible Watermarking". IEEE International Conference on Image Processing, Oct. 2006. pp 2565-2568

[31] H.-T. Huang, Y.-L. Tang "Reversible Data Embedding Based on Histogram Manipulation". Tesis de maestría. *Department of Information Management. Chaoyang University of Technology*. Taiwan. Julio 2005.

[32] Akiyoshi Wakatani. "Digital Watermarking for ROI Medical Images by Using Compressed Signature Image". Proceedings of the 35th Hawaii International Conference on System Sciences - 2002

[33] 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

[34] Santi P. Maity, A. Banerjee, M. K. Kundu, “An Image-in-Image communication scheme and VLSI implementation using FPGA” 2004

[35] Saraju P. Mohanty, R Kumara C, S Nayak, “*FPGA Based Implementation of an Invisible-Robust Image Watermarking Encoder*” G. Das and V.P. Gulati (Eds.): CIT 2004, LNCS 3356 Springer-Verlag Berlin Heidelberg pp. 344–353, 2004

[36] William A. Irizarry-Cruz, “Fpga Implementation of a Video Watermarking Algorithm” Thesis. University Of Puerto Rico. Mayagüez Campus. 2006

[37] S. Barré: Medical Image Samples:
http://barre.nom.fr/medical/samples/
Y MATLAB Central File Exchange:
http://www.mathworks.ch/matlabcentral/fileexchange/loadFile.do?objectId=2762&objectType=FILE

[38] The Canterbury Corpus: http://corpus.canterbury.ac.nz/

[39] http://perso.wanadoo.es/chyryes/glosario/FPGA.htm

[40] http://www.xilinx.com/products/silicon_solutions/fpgas/spartan_series/spartan3e_fpgas/index.htm

[41] Xilinx ISE 9.2i Software Manuals. 2007 Xilinx, Inc. en www.xilinx.com

[42] Trithemius, Steganographie: Ars per occultam Scripturam año 1500. Primera edición impresa: Frankfurt, 1606
http://www.esotericarchives.com/tritheim/stegano.htm

[43] http://www.scycore.com/papers/comp_crime.html

[44] Tábara J.L., Breve Historia de la Criptografía Clásica. www.uam.es/otros/fcmatematicas/Trabajos/Bartolome/Breve_Historia_de_la_Criptografia_Clasica.pdf