INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERIA
MECANICA Y ELECTRICA
UNIDAD PROFESIONAL “ADOLFO LOPEZ MATEOS”

CODIFICADOR REED-SOLOMON EN SOFTWARE

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO EN COMUNICACIONES Y ELECTRONICA

PRESENTA

JESUS ESPITIA JUAREZ

ASESORES:

M. EN C. DAVID VAZQUEZ ALVAREZ

M. EN C. GABRIELA SANCHEZ MELENDEZ

MEXICO D.F. 2012

INSTITUTO POLITECNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERIA MECANICA Y ELECTRICA
UNIDAD PROFESIONAL " ADOLFO LOPEZ MATEOS"

T E M A D E T E S I S

QUE PARA OBTENER EL TITULO DE **INGENIERO EN COMUNICACIONES Y ELECTRÓNICA**
POR LA OPCION DE TITULACION **TESIS Y EXAMEN ORAL INDIVIDUAL**
DEBERA(N) DESARROLLAR **C. JESÚS ESPITIA JUÁREZ**

"CODIFICADOR REED-SOLOMON EN SOFTWARE"

IMPLEMENTAR EN UNA TARJETA DE DESARROLLO FPGA UN MÓDULO DE SOFTWARE QUE REPRESENTE UN CODIFICADOR REED-SOLOMON PARA CÓDIGOS BCH, UTILIZANDO EL CONCEPTO DE RADIO DEFINIDO POR SOFTWARE.

- RADIO DEFINIDO POR SOFTWARE Y SUS APLICACIONES
- CÓDIGOS BCH (BOSE-CHAUDHURI-HOCQUENGHEM)
- LENGUAJES DESCRIPTIVOS DE HARWARE
- DISEÑO, PRUEBAS Y ANÁLISIS DE RESULTADOS DE CODIFICADOR REED-SOLOMON

MÉXICO D.F. A 27 DE SEPTIEMBRE DE 2012

A S E S O R E S

M. EN C. DAVID VÁZQUEZ ÁLVAREZ **M. EN C. GABRIELA SÁNCHEZ MELÉNDEZ**

ESCUELA SUPERIOR DE INGENIERIA MECANICA Y ELECTRICA
ESTADOS UNIDOS MEXICANOS
DEPARTAMENTO DE I.C.E.

M. EN C. DAVID VÁZQUEZ ÁLVAREZ
JEFE DEL DEPARTAMENTO ACADÉMICO DE
INGENIERÍA EN COMUNICACIONES Y ELECTRÓNICA

Índice

LISTA DE FIGURAS

LISTA DE TABLAS

OBJETIVO

Implementar en una tarjeta de desarrollo “Arreglo de Compuertas con Campo Programable”, por sus siglas en ingles FPGA (Field Programmable Gate Arrays) un módulo de software que represente un codificador Reed-Solomon para códigos BCH (Bose Chaudhuri Hocquenghem), utilizando el concepto de Radio Definido por Software.

OBJETIVOS ESPECIFICOS

- Revisión de la arquitectura del radio definido por software y sus diferentes aplicaciones en las comunicaciones móviles actuales.

- Revisión de los códigos BCH.

- Revisión del codificador Reed-Solomon.

- Representación de módulos de codificación en lenguajes de descripción de hardware utilizando herramientas visuales para programación de FPGA’s.

- Realización de pruebas y comprobación de resultados.

JUSTIFICACION

Esta investigación se realiza primordialmente para estudiar la metodología de funcionamiento de un código Reed- Solomon; siendo este una parte esencial de la estructura general de un “Transmisor de Video Digital”, por sus siglas en ingles DVB (Digital Video Broadcast), es decir, efectuar un código modificable y reprogramable, mediante los equipos de radio desarrollados por programas o “Radio Software”, por sus siglas en ingles SDR (Software Defined Radio) produciendo facilidad en su uso y adaptación a las necesidades del usuario.

En la actualidad la investigación referente a la metodología de funcionamiento del codificador Reed-Solomon se hace más competitiva dentro de su ramo y cada vez adoptan más estrategias a fin de garantizar el éxito de los transmisores y receptores de las comunicaciones móviles, lo que me ha permitido conocer de manera más detallada el funcionamiento de dicho código, para desarrollar una implementación óptima y general. Por lo tanto, es necesario dar origen a un proceso más eficiente y compacto el cual genere beneficios expresados en la optimización de los procesos, a diferencia de las generaciones pasadas, a fin de mejorar la calidad, el control de la gestión, la satisfacción y la respuesta a los clientes.

Desde el punto de vista teórico, esta tesis generará introversión y discusión tanto en el conocimiento existente del área investigada, como dentro del ámbito de la programación de dispositivos, ya que de alguna manera u otra, se confrontan teorías (en nuestro caso se analizan tres cuerpos teóricos dentro del Codificador Reed-Solomon: Códigos BCH, Códigos cíclicos y Campos de Galois), lo cual necesariamente conlleva hacer epistemología del conocimiento existente. Dicho lo anterior, es necesario analizar si dentro de la metodología de funcionamiento del codificador Reed-Solomon es factible desarrollar una programación óptima para establecer una mejora en su aplicación por medio del radio definido por software.

El progreso de este proyecto ayudará a mejorar la interoperabilidad de las comunicaciones móviles dentro de los siguientes dispositivos; mencionados por orden de importancia: los transmisores DVB, redes inalámbricas, los radios de campo y dispositivos de almacenamiento. Ofreciendo soluciones adaptables para las comunicaciones cotidianas y para los usuarios de cada país y grupos sociales. Debido, que el DVB está presente alrededor de todo mundo por medio de la televisión, y siendo está considerada como un medio de acceso a la información; se ha suscitado la necesidad de afinar la transmisión de señales. Por tal motivo se hace hincapié en el desarrollo del codificador Reed-Solomon como componente en la estructura del DVB.

Todo lo anterior, se expresara mediante una serie de argumentos basados en procedimientos teóricos y técnicos; diseñando un código Reed- Solomon programable en lenguaje descriptivo de hardware también llamado por sus siglas VHDL (Very High Description Languaje), siendo el lenguaje aceptado para las FPGA´s.

CAPITULO 1 RADIO DEFINIDO POR SOFTWARE Y SUS APLICACIONES.

1.1 Radio Definido por Software

Antes de comenzar a hablar sobre los antecedentes del Radio Definido por Software, habrá que ofrecer una breve explicación sobre que es el radio definido por software, pero sólo para que se tenga una idea de lo qué se refiere el presente capitulo. **Los equipos receptores y transceptores de radiocomunicaciones son equipos constituidos por multitud de componentes electrónicos, los cuales forman circuitos sintonizadores, etapas de frecuencia intermedia, detectores, amplificadores de baja frecuencia, etc., es decir, están constituidos por "Hardware" , el Radio Definido por Software esta conformado por componentes que funcionan por medio de programas en un ordenador, es decir, están constituidos por "Software".**

Considerando lo anterior, habrá que proceder a dar una breve recapitulación histórica sobre el radio definido por software, pues es importante señalar que no es una situación nueva, aunque sigue utilizándose de manera muy extendida.

1.1.1 Antecedentes.

En los años 80´s y 90´s se introdujeron microprocesadores en los equipos receptores y transceptores de radiocomunicaciones, para el control de funciones internas (controles desde teclados y pulsadores), además de añadir nuevas prestaciones (relojes, pantallas informativas, programadores, etc.), de igual forma se introdujo la posibilidad de controlar los equipos de radio desde un ordenador, añadiendo al equipo de radio puertos de comunicación o interfaces para la conexión al ordenador. En estos casos, y usando el software adecuado, es posible controlar desde el ordenador numerosas funciones del equipo de radio, igual o mejor que desde los controles del propio equipo.

También en la década de los 90´s se dio paso a la introducción de chips "Procesadores Digitales de Señal" (DSP) en los equipos modernos de radio permitiendo realizar filtros de paso banda y de supresores de ruidos, mediante técnicas digitales, siendo estos dispositivos mayormente eficaces, e incluso mejor que los realizados con circuitos analógicos.

Los equipos de radio realizados enteramente con componentes electrónicos, los informáticos lo denominan como "Radio Hardware". Cabe mencionar que desde principios de la década del 2000, los radio aficionados como Gerald Youngblood, AC5OG, se dieron a la tarea de investigar y desarrollar un nuevo concepto de equipos de radiocomunicaciones y equipos de radio desarrollados por programas, lo que dio origen al concepto de "Radios Software", en siglas SDR (Software Defined Radio), en lo que la parte hardware (circuitería) es mínima, y la mayor parte de las funciones que se definen

un en equipo de radio se especifican por software (programas) en un ordenador PC o de otro tipo, dotado de una tarjeta de sonido (requisito necesario).

Durante la última década, la tecnología en semiconductores ha perfeccionado su capacidad de rendimiento y costo, de tal modo han emergido dentro de los laboratorios de desarrollo e investigación y laboratorios militares nuevas tecnologías de radio que comienzan a prevalecer. Una de estas tecnologías es el **Radio Definido por Software** o *Software Defined Radio [1].*

1.1.2 Concepto de Radio Definido por Software

El termino **radio** se utiliza mediante la combinación de transmisor (Tx) y receptor (Rx), lo que generalmente se conoce como **transceptor**. Una de las principales características del radio, se encuentra en el intercambio de información, y no solo puede transportar audio, sino eventualmente video y transmisiones multimedia [2].

Un SDR, tiene casi todos sus "Componentes" definidos y funcionando en forma de programas en un ordenador, a excepción de un mínimo de componentes físicos. Y mientras no sea activado ese software o conjunto de programas, el equipo de radio no será utilizado como tal, sino que será un simple conjunto de placas electrónicas externas, incapaces de cumplir con su funcionalidad.

Además, un SDR es muy flexible, ya que modificando o remplazando sus programas de software, se logra modificar sus funcionalidades, como es añadir nuevos modos o mejorar sus beneficios. De esta forma es posible acomodar el SDR a las necesidades de cada tipo de usuario (radioaficionados, servicios de emergencias, etc.). **El SDR es capaz de ser re-programados o reconfigurado para funcionar con diferentes protocolos y formas de onda a través de la carga dinámica de nuevas formas de onda y protocolos.**

Debido a la facilidad y flexibilidad del radio definido por software; me vi en la necesidad de desarrollar y diseñar codificadores, basados en software, para los códigos BCH; también hacer la detección y corrección de errores, esto apoyado en el codificador y algoritmo Reed-Solomon. Utilice la programación de una FPGA para realizar la manipulación de la codificación recibida en la señal.

Dentro del SDR existe otro término importante llamado Radio Cognitivo, por sus siglas en inglés CR (*Cognitive Radio*) el cual se define como un dispositivo que es capaz de extraer la información necesaria del entorno radio y modificar sus parámetros de transmisión con el fin de utilizar de manera eficaz el espectro disponible. La idea básica del Radio Cognitivo es que un usuario secundario, sin licencia, utilice los recursos libres que los usuarios primarios, con licencia, no están utilizando [3].

Así por tanto los conceptos mencionados, concluyo que el SDR es un software dentro de una tarjeta programable que podrá satisfacer las necesidades del usuario que en conjunto del radio cognitivo aprenderá y observara por sí mismo.

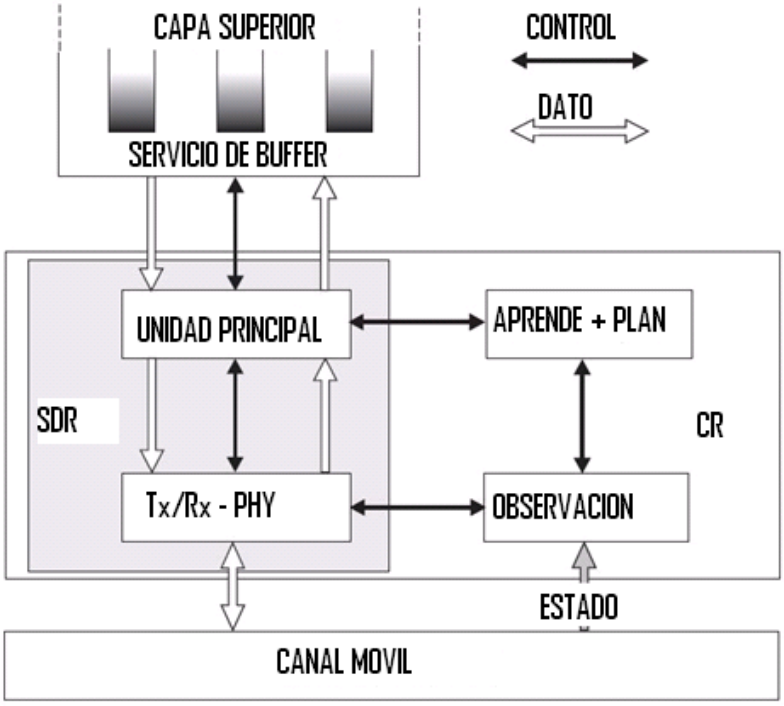


Figura 1-1 Estructura General del Radio Definido por Software y Radio Cognitivo

Con el esquema anterior, lo que intento explicar es la estructura general de Radio Definido por Software y Radio Cognitivo (CR), donde se localizan las funciones de un Radio Software (SR) que son realizadas con programas en ejecución en un procesador adecuado. Los algoritmos (pasos) del canal móvil producen señales de radio compatibles con los estándares que son soportados por SDR (unidad principal). Los algoritmos del receptor son desarrollados óptimamente para recuperar la información enviada por el transmisor (CR).

De acuerdo a su área de operación un SDR puede ser:

- Un sistema *Multibanda:* soporta más de una banda de frecuencias dedicadas, usado para estándares wireless (como GSM 900, GSM 1800, GSM 1900).

- Un sistema *Multiestándar:* soporta más de una interface aérea. Los sistemas *Multiestándar* pueden trabajar dentro de una familia estándar (como UTRA-FDD, UTRA-TDD para UMTS) a través de diferentes redes (como DECT, GSM, UMTS, WLAN).

- Un sistema *Multiservicio:* provee diferentes servicios (Datos, Telefonía, Video).

- Un sistema *Multicanal:* soporta dos o más transmisiones independientes y recibiendo canales simultáneamente.

El Radio Definido por Software tiene algunas características que lo hacen único en comparación con otros tipos de radios. Una de ellas es que tiene la capacidad de ser transformado a través del uso de software o de lógica re definible, a menudo esto se hace con Procesadores Digitales de Señales (DSPs) o con FPGAs (Field Programmable Gate Arrays) de propósito general.

A continuación se presenta de forma esquemática el Transceptor del Radio Definido por Software, con el propósito de tomar ventaja del procesamiento digital de las señales analógicas tradicionales las cuales deben estar convertidas al dominio digital mediante el uso de convertidores Analógico-Digital y Digital-Analógico. Para obtener mejores resultados del procesamiento digital de señales, los SDR deben guardar las señales digitales lo más cerca posible de la antena de tal forma que pueda reconstruir más cadenas de señales digitales. [4]

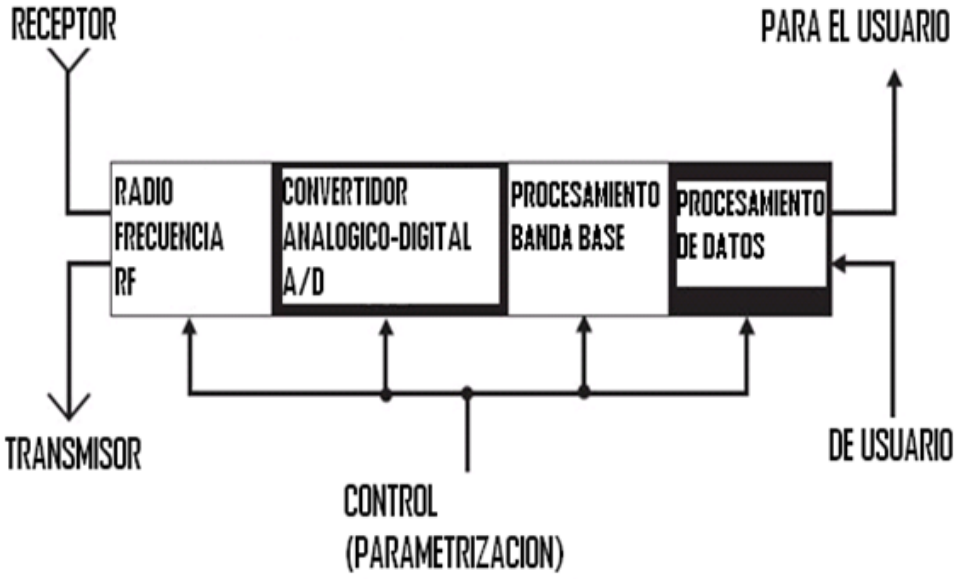


Figura 1-2 Transceptor del Radio Definido por Software

La opción que proporciona la digitalización de señales en la antena, es el uso de un *Analogue Front End (AFE)* flexible, capaz de interpretar un amplio rango de frecuencias y bandas para que con los convertidores de datos puedan procesarlas adecuadamente.

El punto clave es que los SDRs tienen la habilidad de ir más allá de la simple tecnología *single-channel transceiver (Transceptor de Canal Único)* y *single-mode transceiver (Trnsceptor de Modo Único)* con la habilidad para cambiar la función arbitrariamente, ya que el ancho de banda del canal, tasa y modulación son determinados a través de software.

1.2 Aplicaciones en el Digital Video Broadcast (DVB)

El Digital Video Broadcast es una organización que promueve estándares aceptados internacionalmente de televisión digital, en especial para HDTV (High Definition TV) y televisión vía satelital, así como para comunicaciones de datos vía satélite. Por lo tanto permite ejecutar estrategias de inclusión social y reducir la brecha digital, ofreciendo soluciones adaptables para cada país y los grupos sociales, considerada así más que alta definición.

De tal modo, se aprecia que la televisión digital será la única opción de accesos a los servicios de la información para una gran parte de la ciudadanía [5].

Hay una amplia variedad de estos esquemas de modulación en uso y su variedad depende principalmente de tres factores:

- El medio de transmisión:
  - Satélite.
  - Cable.
  - Terrestre.
- La aplicación o servicio.
- El país de implantación [6].

En la siguiente figura se muestra el esbozo de las características y elementos de los diferentes DVB.

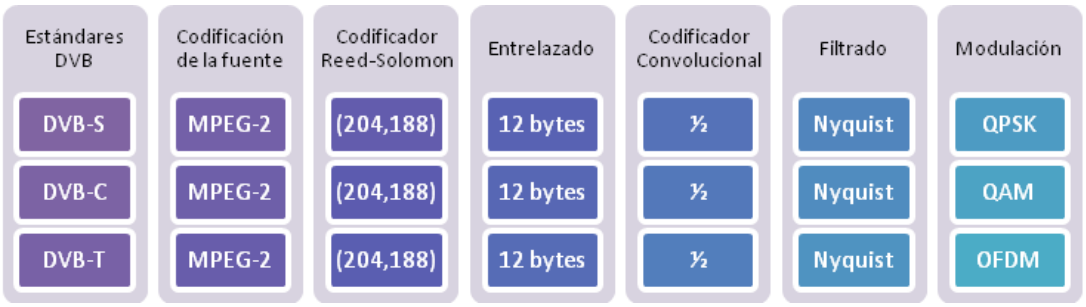


Figura 1-3 Esquema y características de los diferentes elementos DVB

Pero por cuestiones de estudio, hablaré únicamente de los tres primeros elementos que son: Estándares DVB, Codificación de la Fuente y Codificador Reed- Solomon; haciendo mayor énfasis en este último.

Una de las primeras decisiones del DVB fue utilizar el MPEG-2 como estándar de compresión de vídeo y audio, además de definir las técnicas de modulación y métodos de codificación para la corrección de errores, que permitan la transmisión vía satélite, cable y terrestre (DVB-S, DVB-C y DVB-T).

El codificador Reed-Solomon empleado en los estándares DVB es el RS (204,188). Para desarrollar la metodología a emplear en el diseño de codificadores RS, opte por trabajar con el codificador RS (7,3) y así implementarlo en la FPGA y comprobar su funcionamiento.

1.3 Aplicaciones en Comunicaciones Móviles

La tecnología de Radio Definido por Software se creó para mejorar la interoperabilidad entre diferentes redes inalámbricas, las radios de campo, y dispositivos de almacenamiento. Dicha tecnología se compone de software y hardware que pueden ser reconfigurados dinámicamente para permitir la comunicación entre una amplia variedad de comunicaciones en la evolución de las normas, protocolos y enlaces de radio.

De esta manera el SDR, puede crear multi-modos, multi-bandas, multifuncionales dispositivos inalámbricos y equipos de red que pueden ser modificados resolutivamente, mejorado y reestructurado a través de actualizaciones de software y hardware de la reconfiguración.

El Radio Definido por Software tiene soluciones provenientes de Texas Instruments DSP's, que ofrecen los desarrolladores con la flexibilidad para diseñar una variedad de radios de comunicación inalámbrica. Estas soluciones incluyen el estándar de la industria de software y hardware con herramientas de desarrollo que de manera significativa reducen el tiempo de salida al mercado y el costo.

Sundance SMT8096 es la plataforma de desarrollo compatible con el diseño y desarrollo de una amplia gama de aplicaciones de la radio y la forma de onda. Se puede utilizar para prototipos militares y la seguridad pública de radios de comunicación, estaciones de base inalámbricas y para aplicaciones de adquisición de datos de alta velocidad.

Los radios flexibles de cadena ya se pueden encontrar en el mercado. Un ejemplo claro de esto es el PDA (Asistente Digital Personal) donde son compatibles varias normas (WiFi, 3G, Bluetooth, GSM, etc.). Otro ejemplo de presencia en el mercado se encuentra en los productos que brindan Orange Unik que apoyan tanto WiFi y 3G. Estos productos pueden ser definidos como SDR, ya que su banda base de procesamiento pueden ser adaptados a la norma de apoyo deseada. Sin embargo, el concepto de SDR es más amplio.

Unik son los primeros radios cognitivos. De hecho, para reducir la carga de las IMT (GSM o 3G) de redes, los productos son capaces de cambiar a la norma si un WiFi compatible con punto de acceso se detecta a su alcance. La parte cognitiva consiste en detectar estos puntos de acceso WiFi y cambiar la norma de comunicación, si es posible.

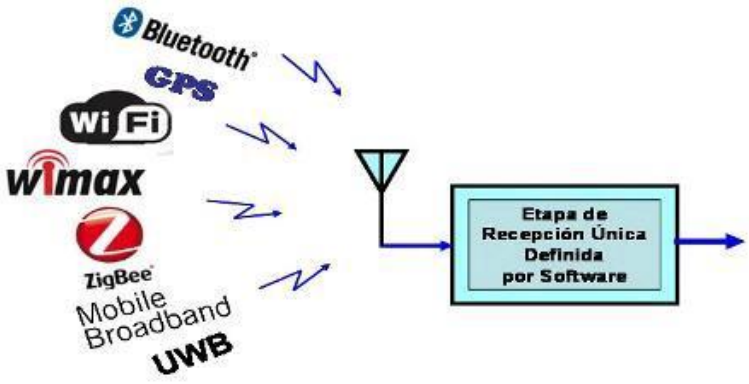


Figura 1-4 Aplicaciones de SDR en el mercado

El concepto de radio cognitiva puede ampliarse a todo tipo de aplicaciones. Por ejemplo, un PDA para apoyar diversas interfaces de comunicación de datos, puede elegir por sí mismo la norma más adaptada a su necesidad y a estos campos electromagnéticos que lo rodee [7].

CAPITULO 2 CODIGOS BCH (BOSE-CHAUDHURI-HOCQUENGHEM) Y CODIGO REED-SOLOMON

Para poder comprender la función de los códigos Reed-Solomon, es necesario saber conceptos y fundamentos para generar este tipo de código, tales como: los códigos BCH (Bose-Chaudhuri-Hocquenghem), los códigos cíclicos, los polinomios de palabra código y detección y corrección de errores; que serán algunos de los temas a tratar en este capítulo.

2.1 Definición de códigos BCH

Los códigos de BCH fueron inventados en 1959 por Hocquenghem, e independientemente en 1960 cerca Bose y Rayo-Chaudhuri. Las siglas *BCH* abarca las iníciales de los nombres de estos inventores [8].

Los códigos BCH (Bose Chaudhuri Hocquenghem) son un subconjunto de códigos cíclicos, de tipo binario o de tipo no-binario (como los Reed-Solomon, por ejemplo)[9] ; el código BCH es el código más conveniente para errores independientes. Los parámetros definidos son:

- Longitud del bloque código:

  $n = 2^m - 1$, donde m>=3.

- Dimensión del código:

  k = n – grado del polinomio generador (gx); el cual depende del número de errores que pueda corregir el código *t*.

En la siguiente tabla se muestra la familia de códigos BCH de longitud 15.

Tabla 1 Código BCH de longitud 15.

n	k	t
15	11	1
15	7	2
15	5	3

La inserción de estos códigos son los siguientes; reciben un paquete de información de longitud *k*, se procesa convirtiéndolo en un bloque de longitud *n*, donde *n* > *k*, esta situación se muestra en la siguiente figura: [10].

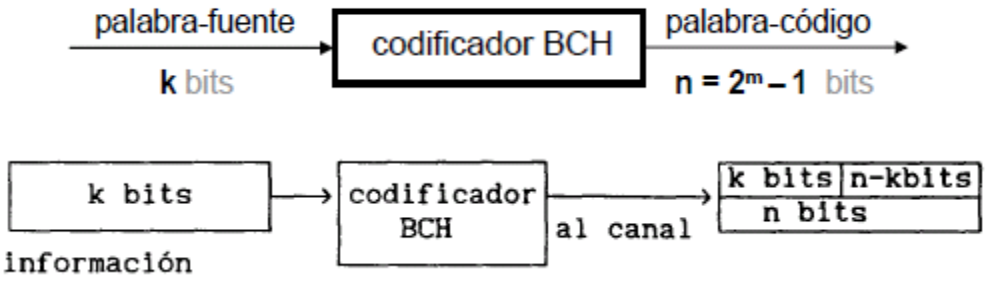


Figura 2-1 Representación de inserción de bits en la información

Para tamaños de unos pocos cientos de bits o menos, los códigos *BCH* son de los mejores para un mismo tamaño de bloque e índice de código (relación entre el bloque de entrada y el bloque de salida). Algunos códigos comunes expresados en la forma (*n, k, t*) de *BCH* son: (7, 4, 1), (15, 11, 1), (15, 7, 2), (15, 5, 3), (31, 26, 1), (31, 21, 2), (31, 16, 3), (31, 11, 5) y (31,6, 7) [12].

Con esto concluyo que los códigos BCH sirven para entrar la explicación de las bases y propiedades del codificador Reed-Solomon.

2.2 Códigos cíclicos

Los códigos cíclicos son una subclase de los códigos de bloque lineales, los cuales tienen esquemas de decodificación eficientes, es decir con algoritmos relativamente simples. Se dice que un código es cíclico cuando cualquier desplazamiento en lazo cerrado de una palabra-código da como resultado otra palabra-código existente dentro del conjunto empleado para codificar los posibles mensajes.

Existen una gran variedad de códigos cíclicos. Por ejemplo, el Código de Redundancia Cíclica empleado en comunicaciones de datos y el código Golay que es un código binario como el Hamming. Además, están los códigos como el Bose-Chaudhuri-Hocquenqhem y el Reed Solomon. Dada la versatilidad de parámetros de estos dos últimos son los que se seleccionaron para ser analizados en esta sección y en las siguientes.

2.2.1 Polinomios de palabra-código

La representación matemática de la operación de los códigos cíclicos está basada en el uso de polinomios. Los elementos de una palabra-código de tamaño *n* pueden ser los coeficientes de un polinomio de grado *n-1*. Por ejemplo, la palabra-código con elementos $x_0, x_1, \ldots, x_{n-1}$ puede ser representada en forma de polinomio como:

$$X(D) = x_0 + x_1 D, \ldots, + x_{n-1} D^{n-1}$$

Dónde: D es una variable Real arbitraria.

2.2.2 Polinomio generador

Un código cíclico (*n, k*) es especificado por un conjunto de polinomios de palabra-código de grado *n*-1 o menos, el cual contiene un polinomio de grado mínimo *n-k* como un factor. Este factor especial, denotado por *g(D)* es seleccionado como el Polinomio Generador del código.

2.2.3 Procedimiento para codificación de un código cíclico

Multiplicar el polinomio del mensaje *m(D)* por D^{n-k}

$$D^{n-k}\, m(D) = m_0 D^{n-k} + m_1 D^{n-k+1} + \ldots + m_{k-1} D^{n-1}$$

Dividir $D^{n-k}\, m(D)$ por el polinomio generador *g(D),* obteniendo el residuo *b(D).*

$$\frac{D^{n-k} m(D)}{g(D)} = a(D) + \frac{b(D)}{g(D)}$$

Agregar *b(D)* a $D^{n-k}\, m(D)$ para obtener el polinomio de la palabra-código *x(D).*

$$x(D) = b(D) + D^{n-k}\, m(D)$$

2.2.4 Cálculo del síndrome para detección de error

Considerando que la palabra-código recibida con error sea:

$$Y(D) = y_0 + y_1 D, \ldots, + y_{n-1} D^{n-1}$$

El polinomio del síndrome se obtiene con el residuo de la división del polinomio de la palabra código entre el polinomio generador *g*(*D*).

$$\frac{Y(D)}{g(D)} = q(D) + \frac{s(D)}{g(D)}$$

Dónde: q es el cociente y s es el síndrome.

De manera similar al código Hamming, con el síndrome y el patrón de errores se hace la detección del error o los errores y su posición dentro de la palabra-código [11].

2.3 Detección y corrección de error

En comunicaciones prácticamente todas las señales digitales producidas en la actualidad llevan asociados el proceso de detección o corrección de errores. Este proceso se ocupa de la detección mediante los métodos CRC y BIP y corrección de errores mediante FEC a bloques y convolucional.

Para detectar que hubo un error, al enviarse un marco se guarda en una tabla cuándo se envió y se le asocia un tiempo para recibir su confirmación. Si no se recibe la confirmación por parte del receptor, se re-envía el marco. El problema que puede surgir es que si se perdió la confirmación, el receptor puede tener marcos duplicados, lo cual se soluciona al asignar un número de secuencia a cada marco, para descartar los duplicados y re-enviar su confirmación.

Otra forma de detectar un error (que ya no fue la pérdida del marco, sino la corrupción de su contenido), es insertar un código de chequeo, y para esta labor se utilizan códigos basados en el concepto de "distancia de Hamming".

La distancia de Hamming para un código cualquiera se define como el número de bits diferentes al hacer un XOR entre todos sus símbolos.

Si los símbolos de un código difieren al menos en 2X+1 bits, al variar X bits (dañar X bits) obtengo un nuevo símbolo que se parecerá más en un bit a un código válido que a otro código válido y por lo tanto puede decir que el símbolo dañado en realidad es el más parecido realizando así su corrección.

Para el diseño estándar de protocolos, se han especificado algunas cadenas de chequeo, bien conocidas como CRC-12, CRC-16 y CRC-CCITT con CRC=12,16 bits y CCITT=16 bits respectivamente. Estas cadenas se interpretan con polinomios de la siguiente manera.

$$\text{CRC-12} = 1100000001111 = X^{12} + X^{11} + X^3 + X^2 + X + 1.$$

$$\text{CRC-16} = 11000000000000101 = X^{16} + X^{15} + X^2 + 1$$

$$\text{CRC-CCITT} = 10001000000100001 = X^{16} + X^{12} + X^5 + 1$$

Se observa que la posición del bit con un uno representa la potencia del polinomio. Cada uno de estos polinomios se conocen como "generador polinomial" y las siglas CRC significan "Cyclic Redundancy Code" (Código de Redundancia Cíclica) [12].

2.3.1 FEC (Forward Error Correction, Corrección de error)

A menudo en las listas de canales se ven números como en este ejemplo:

- TVE 27,500 7/8.
- Canal Tal 11,500 3/4.

Los números al final después del SR en forma de quebrados son el FEC y casi todos los receptores del mercado lo detectan de forma automática, pero no era así en los primeros receptores de los 90 cuando comenzábamos a trabajar en esta industria.

FEC es un término llamado **Forward Error Correction** que es aplicable solo a transmisiones digitales y que es una "Repetición" de ese dato para asegurar que la misma

llegue sin perdidas al receptor o usuario sin que pierda la señal ni su calidad. Para hacerlo más sencillo, es una transmisión "Repetitiva".

Ese FEC indica cuantos bytes se usan para una señal y cuantas correcciones de errores se usan en la misma. Por ejemplo, un FEC de 1/2 significa que 1 byte de cada 2, se usa para control de errores y corregir esos errores; cuando un FEC de 7/8 por ejemplo, significa que 7 de cada 8 se usan para corregir esos errores.

En el mundo de la transmisión digital, un FEC de 1/2 da la posibilidad de una transmisión casi perfecta y sin fallas de recepción porque cada byte de la señal, es controlado por otro byte que la corrige. Pero cuando un proveedor usa 7/8 de FEC por ejemplo, significa que no pierde ancho de banda contra el costo de entregar la señal al recipiente o receptor.

Mientras más bajo el nivel de FEC, mejor equipo de recepción se necesita [13].

2.4 Definición de codificador

Un codificador es un circuito combinacional con dos veces más entradas que salidas, cuya misión es presentar en la salida el código binario correspondiente a la entrada activada.

Existen dos tipos fundamentales de codificadores:

- Los primeros solo admiten una entrada activada, codificando en la salida el valor binario de la misma y cero cuando no existe ninguna activa.
- En los segundos puede haber más de una entrada activada, existiendo prioridad en aquella cuyo valor decimal es más alto [14].

La codificación para control de errores corresponde a una rama de las matemáticas aplicadas llamada teoría de la información. Una aplicación específica corresponde a los códigos Reed-Solomon; los algoritmos que maneja esta aplicación pueden ser implementados tanto en software como en hardware.

En vista de la creciente tendencia hacia el uso de dispositivos de lógica reconfigurable a alta escala de integración y de los beneficios que esta tecnología ofrece a los diseñadores de sistemas digitales, mediante el empleo de un lenguaje de descripción de hardware como VHDL, que permite configurar sistemas digitales según las especificaciones demandadas por los usuarios, ajustar cambios en la programación y optimizar los diseños tratándolos en forma modular, se plantea el diseño de estos módulos de codificación bajo esta tecnología.

Una importante característica de VHDL es su estandarización bajo la norma 1076. Es por ello que se ha seleccionado como lenguaje para la descripción del codificador de canal digital.

2.5 Bases del código Reed-Solomon

En el estudio de los codificadores de canal se presenta el código Reed- Solomon, el cual resulta ser el más ventajoso (véase Figura 2.2). Se puede observar que su probabilidad de error en relación con la señal a ruido está cercana al límite de Shannon y presenta mayor eficiencia sobre otros códigos correctores de error en cuanto a ganancia del código. La clave para hacer del código Reed-Solomon una aplicación tecnológica fue la implementación de un algoritmo eficiente de decodificación desarrollado por Berlekamp.

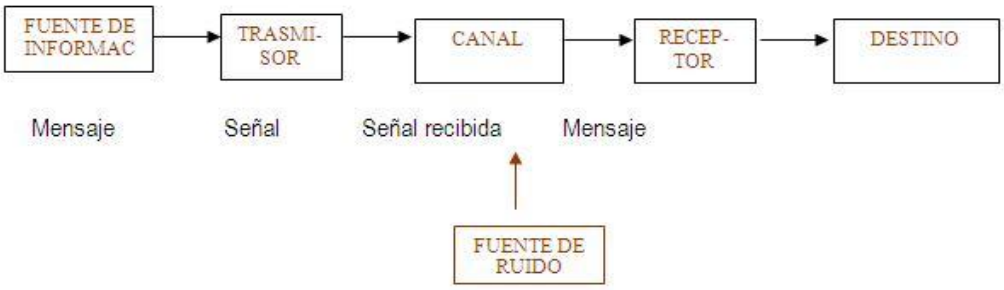


Figura 2-2Modelo de Shannon para la corrección de errores

El código Reed-Solomon es un código corrector de errores basado en bloques en donde el codificador procesa un bloque de símbolos de datos, a los que agrega redundancia para producir un bloque de símbolos codificados.

En la actualidad, los códigos Reed-Solomon se utilizan para corregir errores en varios sistemas incluyendo los dispositivos de almacenamiento –cintas, discos compactos, DVD, códigos de barras, etc.–, comunicaciones inalámbricas o móviles –telefonía celular, enlaces de microondas, etc.–, comunicaciones satelitales, televisión Digital/ DVB, módem de alta velocidad como ADSL, x DSL.

2.6 Propiedades de los códigos Reed−Solomon

El código Reed-Solomon es un subconjunto de los códigos BCH (Bose Chaudhuri Hocquenqhem), códigos cíclicos que presentan entre sus parámetros (*n,k,t*) una relación entre los símbolos de datos (*k*), del código total (*n*) y del número máximo de errores por ser corregidos (*t*), y son de bloques lineales. Un código Reed-Solomon se especifica como RS(n,k) con símbolos de s bits. Lo anterior significa que el codificador toma k símbolos de los s bits y añade símbolos de paridad para hacer una palabra de código de n símbolos.

Existen n-k símbolos de paridad de s bits cada uno. Un decodificador puede corregir hasta t símbolos que contienen errores en una palabra de código, donde 2t = (n-k).

La Figura 2.3 muestra una típica palabra de código Reed-Solomon que se conoce como un código sistemático puesto que los datos se dejan inalterados y los símbolos de paridad se anexan.

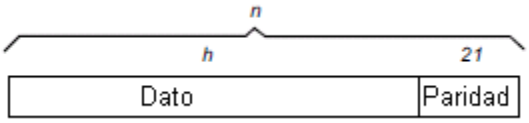


Figura 2-3 Palabra de código Reed-Solomon

Para codificar la trama con esta estructura se debe procesar a través de un circuito digital que opere bajo los fundamentos de campo finito de Galois. Este presenta una arquitectura en el codificador compuesta por los bloques funcionales mostrados en la Figura 2.4.

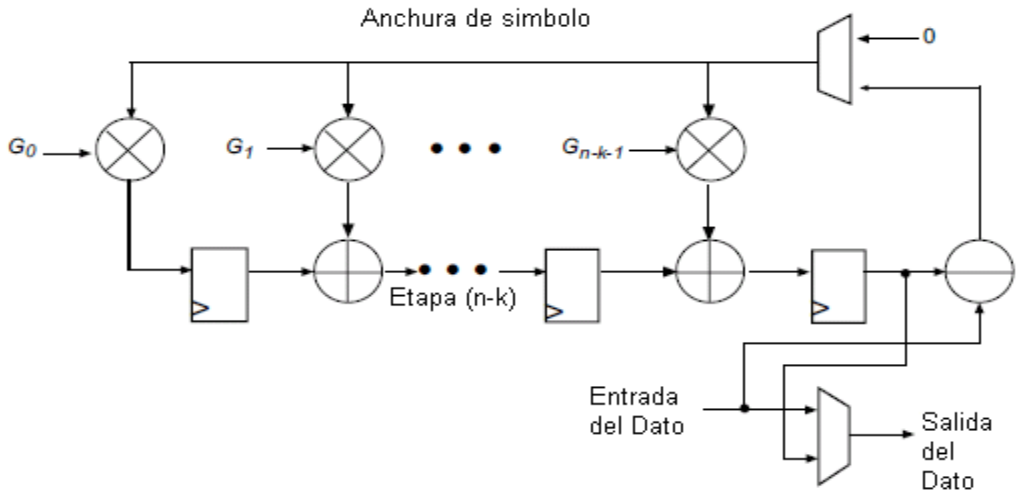


Figura 2-4 Arquitectura genérica de un codificador Reed-Solomon

2.7 Campos de galois aplicados a la codificación Reed−Solomon

Los códigos Reed-Solomon se basan en un área especializada de la matemática llamada campos de Galois o campos finitos. Un campo finito tiene la propiedad de que las operaciones aritméticas sobre elementos del campo siempre tienen un resultado en el campo. Un codificador o decodificador Reed-Solomon debe ser capaz de realizar estas operaciones aritméticas.

2.8 Generador polinomial Campos de Galois

Una palabra de código Reed-Solomon es generada usando un polinomio especial. Todas las palabras de código válidas son divisibles exactamente por el polinomio generador representado por la siguiente ecuación.

$$g(x) = (x - \alpha^{i})(x - \alpha^{i+1})(x - \alpha^{i+2t})$$

La palabra de código se genera de c(x)=g(x)*i(x), donde g(x) es el polinomio generador, i(x) es el bloque de información, c(x) es una palabra de código válida y alfa se conoce como un elemento primitivo del campo.

El primer paso corresponde a la definición del campo de Galois para la codificación, el cual estará definido en función de la longitud del símbolo - entiéndase m, bits/símbolo -, permitiendo así conocer el polinomio irreducible del campo GF(2m), tal que para m=3 bits, corresponde a p(x)= x3+x+1, 1 de manera que las operaciones que produzcan resultados que se rebasan, serán reajustados por el polinomio irreducible, como lo es el caso α3 = α + 1, dando así un elemento perteneciente al campo.

Las bases teóricas que sustentan este codificador están dadas por el polinomio en su forma general.

$$g(x) = \prod_{i=0}^{n-k-1} \left(x - \alpha^{hx(Comienza\ generador+i)}\right)$$

Al expandir el polinomio se obtiene la ecuación siguiente.

$$g(x) = G_{n-k-1}x^{n-k-1} + G_{n-k-2}x^{n-k-2} + \cdots + G_1 x + G_0$$

Dónde:

N: longitud de la palabra codificada (en símbolos).

K: longitud del mensaje codificado (en símbolos).

M: longitud del símbolo (bits) [15].

El segundo paso corresponde a definir el polinomio generador donde se obtiene:

$$G(x) = \alpha^2x^3 + \alpha^5x^2 + \alpha^5x + \alpha6$$

Tabla 2 Multiplicación en el campo finito de Galois (8) para el polinomio generador p(x)=x³+x+1.

	x	000 0	001 1	010 x	011 $x+1$	100 x^2	101 x^2+1	110 x^2+x	111 x^2+x+1
000	0	0	0	0	0	0	0	0	0
001	1	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
010	x	0	x	x^2	x^2+x	$x+1$	1	x^2+x+1	x^2+1
011	$x+1$	0	$x+1$	x^2+x	x^2+1	x^2+x+1	x^2	1	x
100	x^2	0	x^2	$x+1$	x^2+x+1	x^2+x	x	x^2+1	1
101	x^2+1	0	x^2+1	1	x^2	x	x^2+x+1	$x+1$	x^2+x
110	x^2+x	0	x^2+x	x^2+x+1	1	x^2+1	$x+1$	x	x^2
111	x^2+x+1	0	x^2+x+1	x^2+1	x	1	x^2+x	x^2	$x+1$

Esto se utiliza para generar el campo de Galois para el código. Entra como un número decimal donde los bits del archivo binario equivalente corresponden a los coeficientes del polinomio. Por ejemplo:

x8 + x4 + x3 + x2 +1 => 100011101 => 285

Un valor de cero causa al polinomio por defecto el Ancho del Símbolo dado para ser seleccionado. Si el campo polinomial no es primitivo, la siguiente tabla muestra el campo polinomial por defecto.

Tabla 3 Polinomios y tamaño de bits.

Ancho de Símbolo	Polinomio	Representación del arreglo binario	Rango de bits a codificar (k)	Rango de bits codificados (n)
3	x^3+x+1	[1011]	2-5	4-7
4	x^4+x+1	[10011]	2-13	4-15
5	x^5+x^2+1	[100101]	2-29	4-31
6	x^6+x+1	[1000011]	2-61	4-63
7	x^7+x^3+1	[10001001]	2-125	4-127
8	$x^8+x^4+x^3+x^2+1$	[100011101]	2-253	4-255
9	x^9+x^4+1	[1000010001]	2-509	4-511
10	$x^{10}+x^3+1$	[10000001001]	2-1021	4-1023
11	$x^{11}+x^2+1$	[100000000101]	2-2045	4-2047
12	$x^{12}+x^6+x^4+x+1$	[1000001010011]	2-4093	4-4095

CAPITULO 3 LENGUAJES DESCRIPTIVOS DE HARDWARE

En este capítulo entenderemos que en la electrónica, el lenguaje descriptivo del hardware (HDL) es una clase de lenguaje de programación para la descripción formal de circuitos electrónicos. El HDL puede describir la operación del circuito, su diseño y estructura, asimismo verificar las pruebas de su operación por medio de la simulación.

3.1 Antecedentes.

A continuación se presenta una tabla en la que se explican los principales acontecimientos del HDLs: [16]

Tabla 4 Principales acontecimientos del HDLs.

Años	Aspectos
1960s – 1980s	Diferentes lenguajes HDL.
1983	"VHSIC Program" inicia la definición del VHDL.
1987	VHDL Standard (IEEE 1076) aprobado.
1990	Verilog domina el mercado: VHDL empieza a ganar aceptación debido a la estandarización.
1992	Se crea el IEEE 1164
1993	VHDL (revisión): se incorporan cambios menores para facilitar su uso.
Desde 1994	El VHDL ha sido ampliamente aceptado: Todas las herramientas EDA (Electronic Design Automation) de Synopsys, Cadence, Mentor…soportan la especificación, simulación y síntesis con VHDL.

Hacia fines de los 80 mientras los diseños de circuitos integrados de aplicación específica (ASICS) crecían en complejidad los sistemas gráficos para describir estos empezaban a mostrar sus limitantes. El desarrollo, la visualización, depuración y mantenimiento de estos sistemas era cada vez más complicado ya que mientras los diseños alcanzaban las 5000 compuertas lógicas, la cantidad de hojas de esquemas gráficos crecían de igual forma.

El seguir planos esquemáticos con decenas de hojas daba paso a que surjan muchos errores y que el proceso se vuelva muy lento para depurar estos errores [12].

Los primeros HDL modernos son, Verilog, fue introducido en 1985 por una compañía llamada Automated Integrated Design Systems (renombrada después como Gateway

Automation en 1986). El HDL-simulador se convirtió en el estándar de los simuladores de Verilog. En 1987, una petición del departamento de la defensa de E.U.A. produjo el desarrollo de VHDL (Lenguaje Descriptivo de muy alta Velocidad del Hardware).

En la actualidad, VHDL y Verilog emergieron como los HDLs dominantes en la industria de la electrónica, mientras que los HDLs más viejo y menos capaces desaparecieron gradualmente de uso. Pero VHDL y Verilog comparten muchas de las mismas limitaciones: Ninguno de los dos HDL son convenientes para la simulación de circuitos analógicos. Ni uno ni otro posee construcciones del lenguaje para describir las estructuras recurrente-generadas de la lógica.

En breve, se enlista los HDL's más viejos y menos capaces que fueron desapareciendo con el tiempo.

- AHDL (Altera HDL, una lengua propietaria de Altera).
- Átomo (síntesis del comportamiento y HDL de alto nivel basados en Haskell).
- Bluespec (HDL de alto nivel basado originalmente en Haskell, ahora con una síntaxis basado Verilog).
- JHDL (basado en Java).
- MyHDL (basado encendido Python).
- RHDL (basado en Lenguaje de programación de rubíes) [13].

3.2 FPGA: Dispositivo Programable basado en VHDL.

Las FPGA's (Field Programmable Gate Array) son dispositivos lógicos cuyo propósito general es ser programado por los usuarios, compuesto de bloques lógicos comunicados por conexiones programables.

Una vez desarrollado el concepto de FPGA´s, es indispensable señalar las diferentes tarjetas programables que han sido utilizadas a lo largo de esto años, comenzando por la memorias PROM.

La memoria PROM (*Programmable Read Only Memory, Memoria de Solo Lectura Programable*) fue uno de los primeros dispositivos que cumplió con estas características (véase Figura 3.1).

El PLD, (Programmable Logic Device, Dispositivos de Lógica Programable), es una matriz de puertas AND conectada a otra matriz de puertas OR más biestables. La versión más básica del mismo es una PAL, con un plano de puertas AND y otro fijo de puertas OR. Las salidas de estas últimas se pueden pasar por un biestable en la mayoría de los circuitos del mercado (véase Figura 3.2).

La FPLA, (Field Programmable Logic Array, Arreglo de Lógica de Campo Programable) es más flexible que la PAL: se pueden programar las conexiones entre los dos planos. Estos dispositivos son muy simples y producen buenos resultados con funcionalidades sencillas (solo combinacional) (véase Figura 3.3).

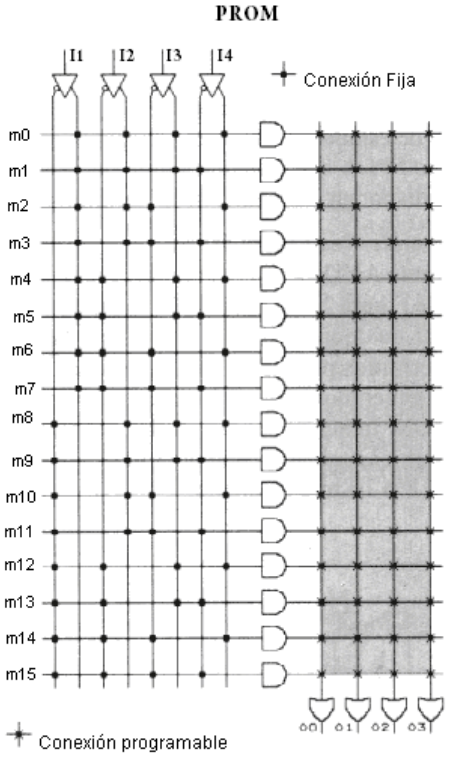


Figura 3-1 Diagrama Estructural de una memoria ROM Programable (PROM)

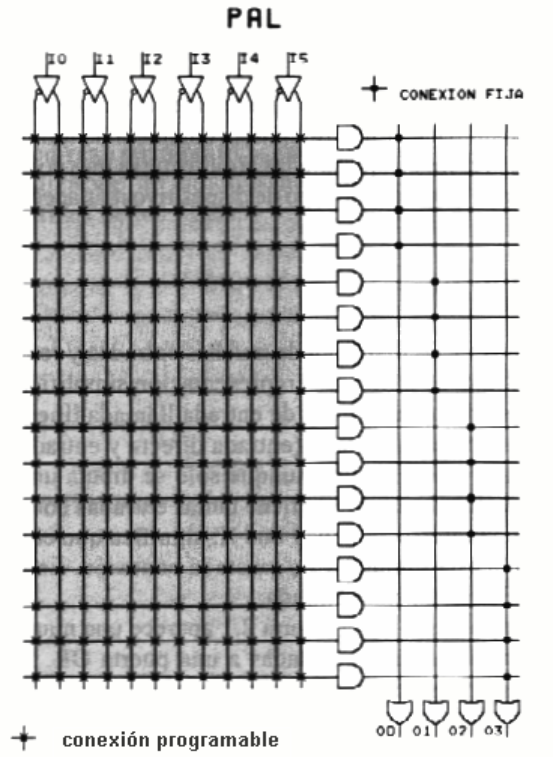


Figura 3-2 Diagrama Estructural de una PAL

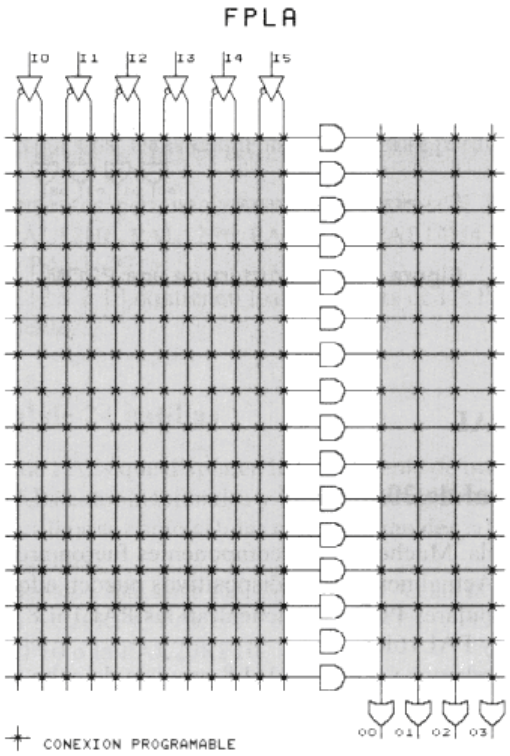


Figura 3-3 Diagrama Estructural de una FPAL

MPGA (Mask-Programmable Gate Array, Arreglo de Mascaras de Compuertas Programable), cuyo principal representante está constituido por un conjunto de transistores más circuitería de E/S. Se unen mediante pistas de metal que hay que trazar de forma óptima, siendo ésta la máscara que hay que enviar al fabricante.

Las FPGA (Field Programmable Gate Array, Arreglo de Compuertas con Campo Programable), introducidas por Xilinx en 1985, son el dispositivo programable por el usuario de más general espectro. También se denominan LCA (Logic Cell Array, Arreglo Lógico de Celdas). Consisten en una matriz bidimensional de bloques configurables que se pueden conectar mediante recursos generales de interconexión. Estos recursos incluyen segmentos de pista de diferentes longitudes, más conmutadores programables para enlazar bloques a pistas o pistas entre sí. En realidad, lo que se programa en una FPGA son los conmutadores que sirven para realizar las conexiones entre los diferentes bloques, más la configuración de los bloques [17].

La gran ventaja de utilizar estos dispositivos radica en que todo el desarrollo se lleva a cabo en un solo ambiente de trabajo. El diseñador propone la función lógica a realizar y en base a métodos de descripción define los parámetros de su problema.

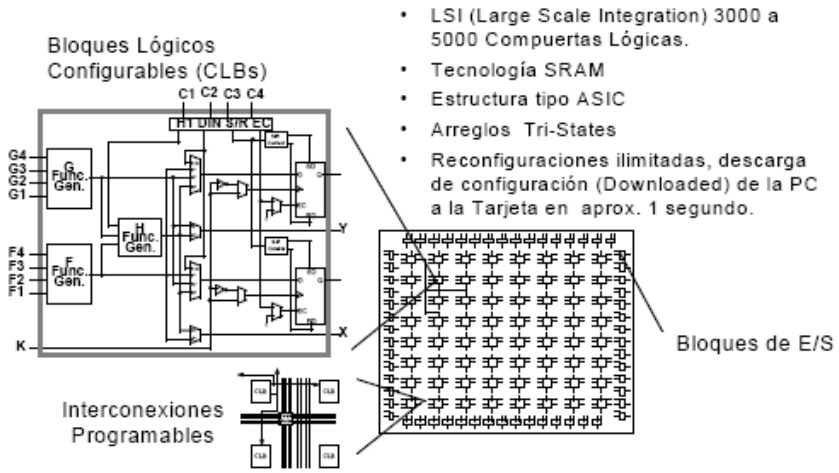


Figura 3-4 Arquitectura de una FPGA

Esto se hace por medio de código programable, que puede ser un Lenguaje de Descripción de Hardware, o bien, un diagrama esquemático de conexiones. Después de la programación se puede ejercer una optimización si se cree necesario hacer modificaciones en el diseño. Sólo bastaría con hacerlas en el código y repetir los pasos anteriormente descritos. El dispositivo se podrá reconfigurar nuevamente hasta en unos 200 intentos, que según el fabricante, es el tiempo de vida promedio [18].

3.3 VHDL (Very High Hardware Description Language)

VHDL es un lenguaje de descripción y modelado, diseñado para describir (en una forma que los humanos y las máquinas puedan leer y entender) la funcionalidad y la organización de sistemas hardware digitales, placas de circuitos y componentes.

Este es un lenguaje con una sintaxis amplia y flexible que permite el modelado estructural, en flujo de datos y de comportamiento hardware. Además permite la configuración precisa, en distintos estilos, del comportamiento de un sistema digital conocido y el desarrollo de tipos de simulación.

La síntesis a partir de VHDL constituye hoy en día una de las principales aplicaciones del lenguaje con una gran demanda de uso. Las herramientas de síntesis basadas en el lenguaje permiten en la actualidad ganancias importantes en la productividad de diseño.

Existen dos formas de describir un circuito. Por un lado se puede describir un circuito indicando los diferentes componentes que lo forman y su interconexión; la otra forma consiste en describir un circuito indicando lo que hace o cómo funciona, es decir, describiendo su comportamiento.

El VHDL permite los dos tipos de descripciones:

- Estructura: VHDL puede ser usado como un lenguaje de Netlist normal y corriente donde se especifican por un lado los componentes del sistema y por otro sus interconexiones.
- Comportamiento: VHDL también se puede utilizar para la descripción comportamental o funcional de un circuito. Esto es lo que lo distingue de un lenguaje de Netlist. Sin necesidad de conocer la estructura interna de un circuito es posible describirlo explicando su funcionalidad. Esto es especialmente útil en simulación ya que permite simular un sistema sin conocer su estructura interna, pero este tipo de descripción se está volviendo cada día más importante porque las actuales herramientas de síntesis permiten la creación automática de circuitos a partir de una descripción de su funcionamiento [5].

Existen programas tanto para Windows como para Linux, para su programación, compilación y simulación, estos son.

- Para Windows: Simulación ModelSim y programación (síntesis) Xilinx ISE
- Para Linux: Xilinx ISE.

3.3.1. Proceso del diseño de una FPGA

En el proceso de diseño de una FPGA, en primer lugar se tiene que especificar el objetivo por el cual se va a realiza la programación exponiendo todos los puntos de funcionalidad; se continua con la descripción de VHDL siendo este el desarrollo de la programación y así poder simular la funcionalidad.

Por otra parte, la síntesis es el arreglo de compuertas que componen la programación creada para dicho objetivo, para posteriormente pasar a la implementación del proyecto que es observar la funcionalidad en tiempo de simulación.

Otro de los aspectos importantes dentro de este proceso es la configuración de la FPGA que de acuerdo al funcionamiento específico se le debe proporcionar los parámetros de misión y hacer la comprobación de los resultados sobre el chip.

Este es el esquema del proceso de una FPGA [16].

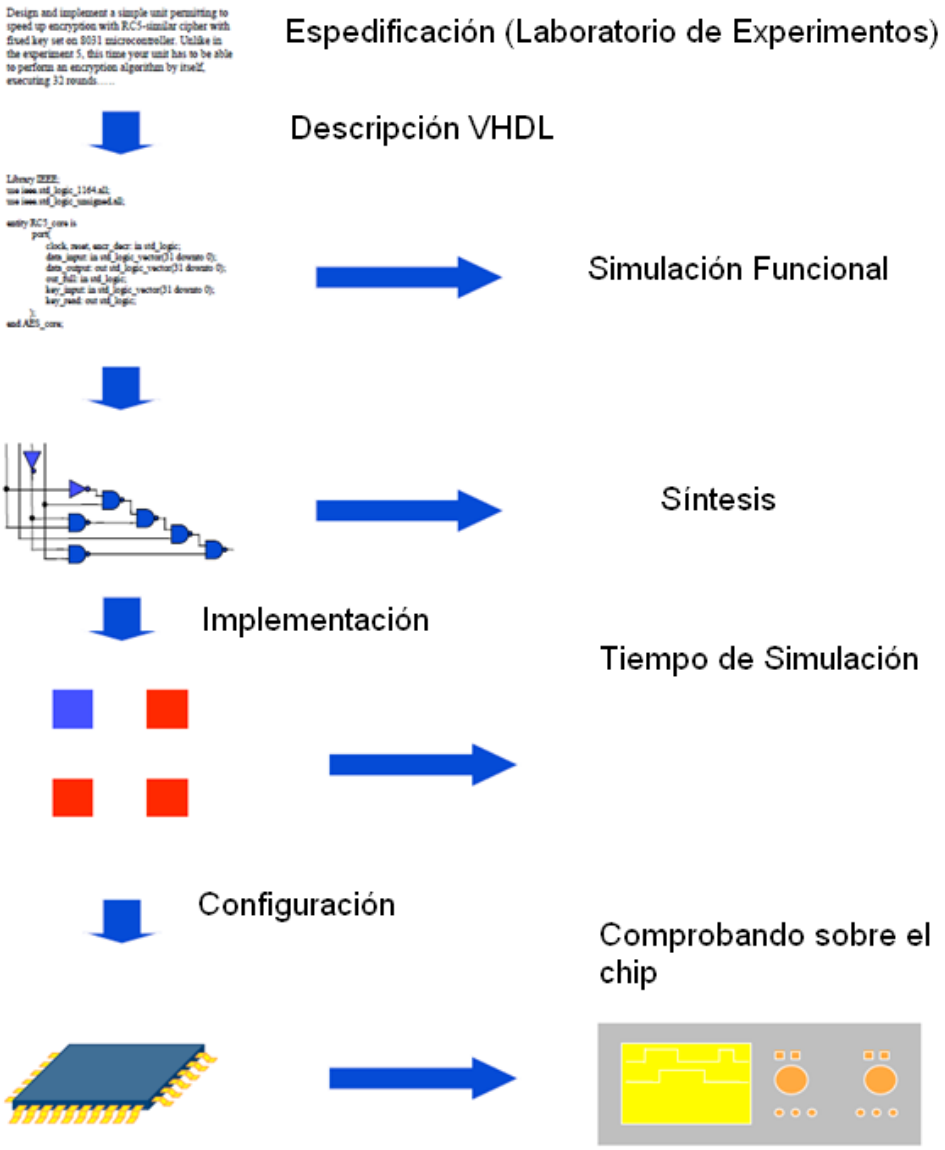


Figura 3-5 Proceso de diseño de una FPGA

CAPITULO 4. DISEÑO, PRUEBAS Y ANÁLISIS DE RESULTADOS DE CODIFICADOR REED-SOLOMON.

Se ha presentado a lo largo de este trabajo, una investigación sobre los elementos que constituyen la base teórica del proyecto de "Implementación de códigos Reed-Solomon para aplicaciones de radio definido por software", con el propósito de entender los motivos del surgimiento de éste tipo de tecnología, así como tener una idea de su funcionamiento y la aplicación en la que se direcciona, a su vez también se refirió la forma en la que se busca implementarla.

Los temas desarrollados en los capítulos anteriores, fueron realizados para una mejor comprensión respecto a las intenciones del proyecto. Ahora toca el turno de hablar sobre las cuestiones prácticas del Codificador Reed-Solomon, haciendo mayor énfasis en el diseño del codificador Reed-Solomon (7, 3) y el diseño de software VHDL para realizar el funcionamiento y poder ser así considerados la parte medular del proyecto.

No obstante, poner a prueba y comprobar su funcionalidad del codificador Reed-Solomon, me proporciona la certeza y la validación que se necesita para sustentar esta tesis. Este proceso se ve reflejado en la Figura 4.1.

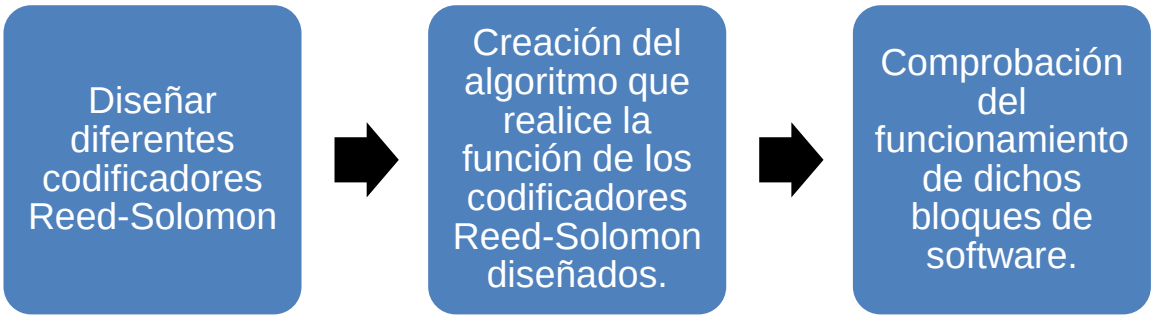


Figura 4-1 Proceso de elaboración de los códificadores Reed-Solomon

4.1 Diseños de codificadores Reed-Solomon.

Para simular el diseño del codificador de Reed-Solomon en el ISE 10.1 de Xilinx se necesita System Generator, el cual proporcionará todas las herramientas para la simulación del mismo.

Los bloques a utilizan son los siguientes:

Bloque Generador Binario de Bernoulli: El bloque Generador binario de Bernoulli genera números binarios aleatorios usando una distribución de Bernoulli.

La distribución de Bernoulli con parámetro P produce 0 (cero) con probabilidad p y 1 (uno) con una probabilidad de 1-p. La distribución de Bernoulli tiene valor medio 1-P y varianza P(1-p). La probabilidad de un parámetro cero especifica p, y puede ser cualquier número real entre cero y uno. El bloque gráfico se muestra en la figura 4.2.

Los atributos de la señal de salida pueden ser un cuadro basado en una matriz, una muestra basada en filas o columnas, o una muestra que se basa en matriz unidimensional. Estos atributos son controlados por el marco base de resultados, las muestras por cuadro, y el vector de interpretación de parámetros 1-D.

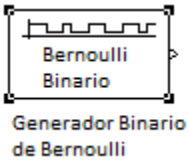


Figura 4-2 Bloque Generador Binario de Bernoulli

Una vez teniendo el generador del mensaje se requiere una compuerta de entrada para introducir nuestros datos al codificador Reed-Solomon.

Compuerta de entrada: compuerta de entrada de Xilinx; son las entradas para los bloques de Xilinx que es parte del diseño de Simulink. Este bloque convierte los tipos de datos a enteros, dobles o de punto fijo dentro del System Generator. Cada bloque define un nivel superior de entrada introducido en el diseño generado por System Generator.

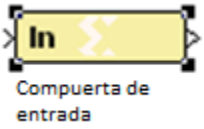


Figura 4-3 Bloque Compuerta de Entrada de Xilinx

Estos dos elementos están conectados en serie hacia el codificador Reed-Solomon en el pin de entrada DATA_IN. Además de esta entrada, se requiere obligatoriamente, las entradas del bypass y start.

En el Bypass se conecta una constante de 0 (cero), ya que no se pretende que el mensaje pase sin ser afectado. Esta constante puede ser tomada simplemente del simulador de Matlab pero seria necesario conectar otra compuerta; para su simplificación tomamos una constante de Xilinx para conectar directamente con el codificador, en la figura 4.4 se muestra la constante de Xilinx.

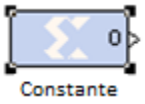


Figura 4-4 Constante 0 (cero) de Xilinx

En el caso de pin de entrada del Start (véase Figura 4.7); se conecta un tren de pulsos para su activación y su sincronización con el mensaje para su codificación de cada bloque.

Se puede generar este pulso con una conexión en serie de una constante en alto, un muestreo en bajo y un inversor. La función del muestreo en bajo es que retarde la constante, en un rango de 0 a 2 sobre el eje de la frecuencia; aun no se genera el pulso pero si se le aplica un inversor tendremos lo siguiente.

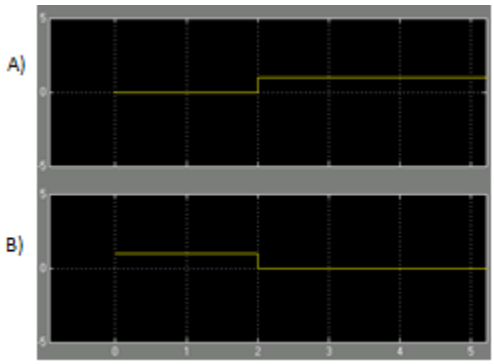


Figura 4-5 A) Constante con el muestreo en bajo y B) Pulso realizado por la constante

Estas conexiones quedan de la siguiente forma en el System Generator:

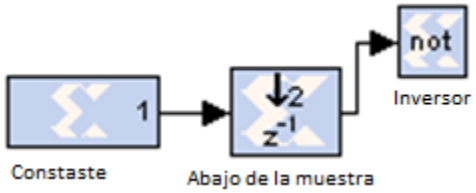


Figura 4-6 Componentes conectados para generar el pulso

El tren de pulsos se realiza con una retroalimentación del pin de salida INFO del codificador Reed-Solomon; pero debido a su funcionamiento no se puede hacer una conexión directa, ya que debe de llevar un convertidor de tipo de variable booleano (falso, verdadero) para así conectar un inversor que genere los pulsos.

Estos pulsos controlaran toda la entrada del mensaje para poder crear los bloques codificados. El pulso de inicio y el tren de pulsos se unen con una expresión lógica OR bit a bit, resultando de la siguiente manera:

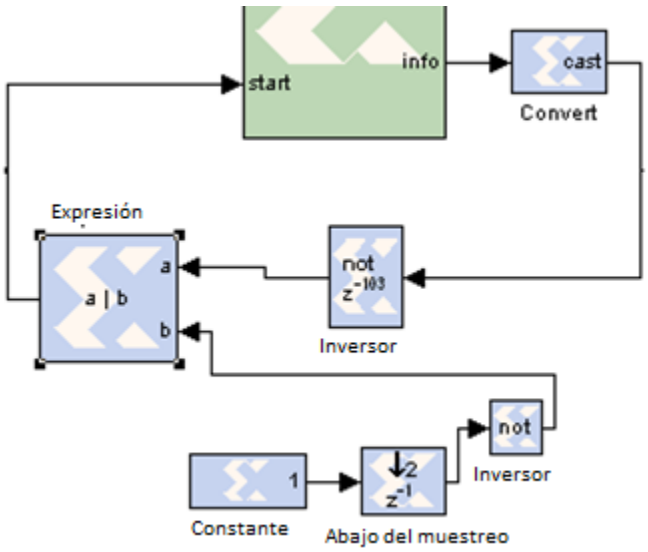


Figura 4-7 Generador de tren de pulsos la entrada de Start

La latencia (retraso) en el inversor se ejecuta para tomar la medida de bloques de símbolos, así mismo llevar el control de los bloques en la salida (DATA_OUT).

Ahora se tiene completo el codificador Reed-Solomon, expuesto en la Figura 4.8; las especificaciones con respecto a la versión de cada codificador se mencionaran a continuación.

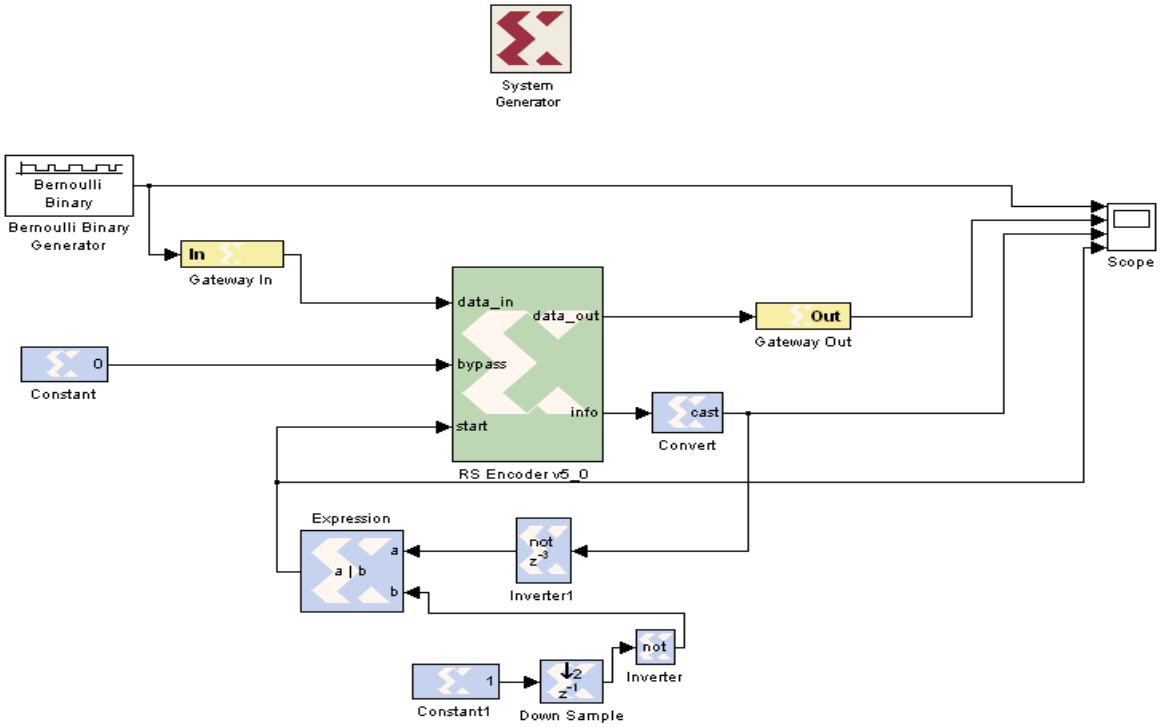


Figura 4-8 Bloques de System Generator del Codificador Reed-Solomon v5.0

4.2 Creación del algoritmo de los diferentes codificadores Reed-Solomon.

Los algoritmos fueron desarrollados en las siguientes plataformas de software: C++, MATLAB y XILINX ISE 10.1, el primero es el entorno común de C o C++ que me facilita el entendimiento del funcionamiento; se sabe que es una forma hibrida y de introducción de datos manual. El MATLAB me brinda un entorno en el cual se pude simular y comprobar el funcionamiento de los codificadores con las funciones propias de éste software, en el tercero se puede implementar los diferentes algoritmos ya probados en MATLAB pero ahora programados en lenguaje de descripción de hardware, el cual es utilizado para la programación de las FPGA, además de que el XILINX ISE 10.1 me proporciona de igual forma herramientas para simular los bloques de software que se han programado y observar el funcionamiento de estos. El proceso del diseño en software de un codificador Reed-Solomon (7, 3) se representa en el diagrama de flujo de la Figura 4.10, el cual puede hacerse general para el diseño en software de cualquier tipo de codificador Reed-Solomon; debido a que se realizan subrutinas, se nuestra en la Figura 4.9 la subrutina de los campos de Galois correspondientes al codificador Reed-Solomon (7, 3).

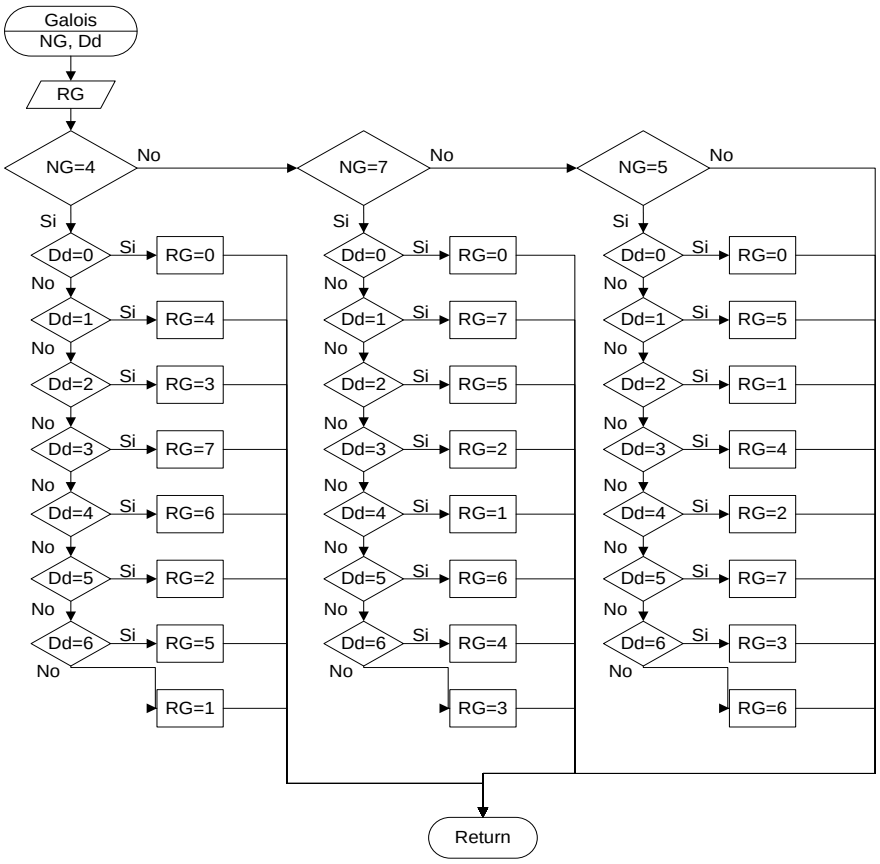


Figura 4-9 Diagrama de flujo de la subrutina de los Campos de Galois

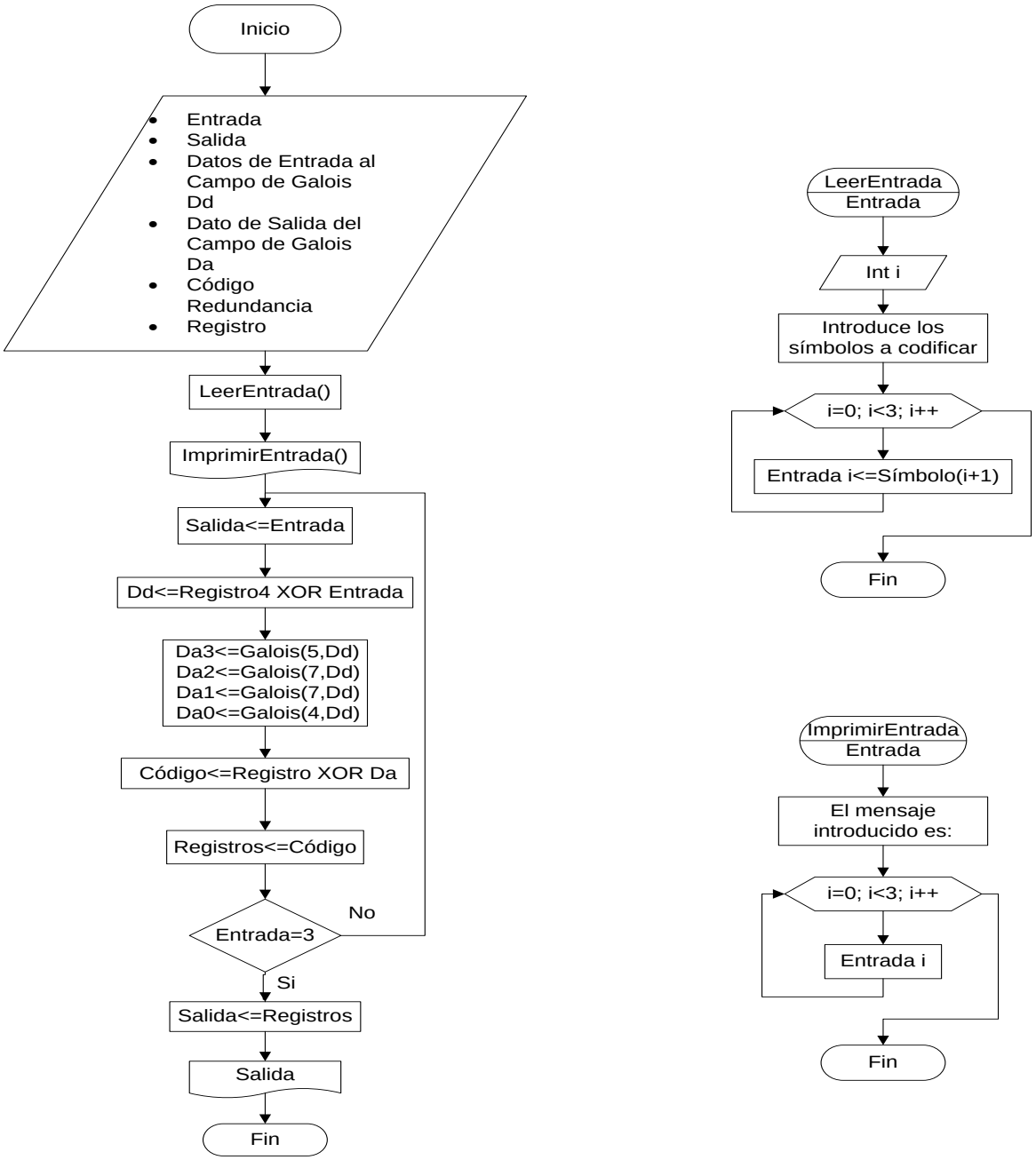


Figura 4-10 Diagrama de flujo para el diseño en software de un codificador Reed-Solomon (7, 3)

Cabe mencionar que éste diagrama de flujo fue implementado en c++ y MATLAB; de tal forma también es aplicable para la programación en el lenguaje de descripción de hardware tal como lo es VHDL, pero obedeciendo los términos y normas para emplearlo en este trabajo.

Los resultados arrojados por las operaciones de las x-or binarias son almacenados en registros que serán vaciados después de haber pasado los 3 símbolos de entrada y así formar la palabra código, ya que éste codificador tiene una relación de codificación RS(7,3) formará por cada 3 símbolos de entrada una palabra código de 7 símbolos.

4.3 Comprobación del funcionamiento de los bloques en MATLAB

4.3.1. Obtención de la información a codificar.

Para la comprobación del funcionamiento de los bloques de software que realizan el funcionamiento de los diferentes codificadores Reed-Solomon se debe obtener la información que se desea codificar.

En donde se ingresa al programa de MATLAB la Figura 4.11 la cual me proporcionará la información a codificar.

Figura 4-11 Imagen para la fuente de información con dimensiones de 102 x 113 pixeles

Originalmente, al ser ingresada la Figura 4.11 en MATLAB, me presenta una matriz de 2 dimensiones de 102 x 113, esto es debido a que se trata de una imagen en blanco y negro.

Dicha matriz me da valores de 0 y 255, estos valores son consecuencia del formato de colores RGB (Red, Green, Blue), el cual es utilizado en el procesamiento de imágenes para indicar todos los tonos de colores que puede tener una imagen. Por tal motivo, la imagen debe de cambiarse a formato binario o mejor dicho, binarizarse; ya que se requiere trabajar solamente con valores lógicos (0 y 1), éste proceso se puede observar de manera más detallada en la Figura 4.12 en donde se ha binarizado una pequeña parte de la figura original. Éste proceso de binarización me facilita una matriz de 102 x 113 elementos que toman valores de 0 y 1, la cual es almacenada. Enseguida la matriz de elementos binarios se manipula para convertirla en un vector, lo anterior para simular una fuente de información continua y binaria, puesto que los codificadores Reed-Solomon trabajan con este tipo de información.

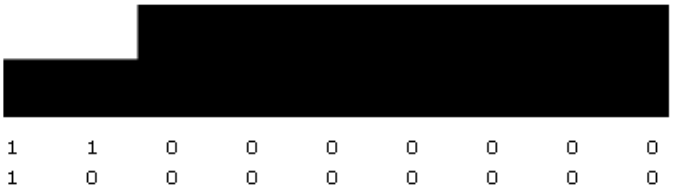


Figura 4-12 Muestra de la imagen binarizada y su código binario

Una vez binarizada la imagen se realiza el proceso de codificación Reed-Solomon antes mencionado en el diagrama de flujo de la Figura 4.10. En la Figura. 4.13, se puede observar la imagen que ha sido procesada por el codificador Reed-Solomon (7, 3) después de haber ingresado la imagen binarizada como la información a codificar.

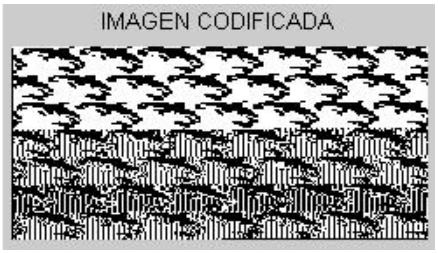
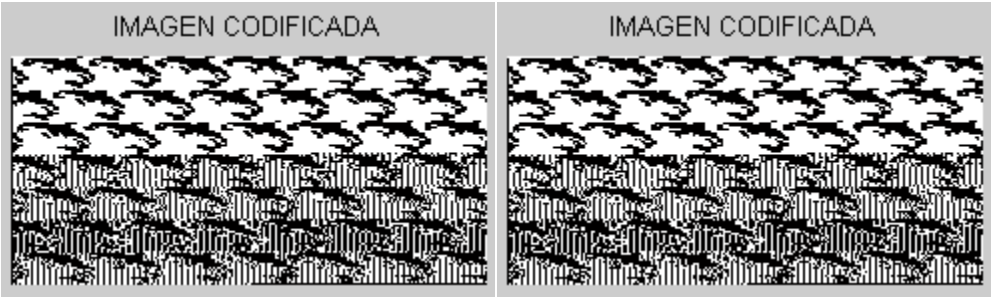


Figura 4-13 Imagen codificada por el codificador Reed-Solomon (7, 3)

Como se puede observar la imagen fue modificada en el proceso de codificación de canal, y de éste modo la información codificada se puede adaptar en el siguiente bloque de modulación para ser enviada a través del canal de comunicaciones. Cabe mencionar que para poder presentar la imagen codificada se tuvo que hacer un arreglo al vector de la información que salió del codificador, ya que, al igual que la información de entrada del codificador ingresa de forma continua, la información codificada (palabras código) sale de forma continua.

4.3.2. Resultados.

Los resultados que se obtuvieron al realizar las pruebas correspondientes a cada codificador Reed-Solomon diseñado, fueron que, el codificador Reed-Solomon propio de MATLAB fue compatible con el diseño, no importando que tuviera una base de programación diferente ya que aceptó la información de la imagen introducida en la Figura 4.11; el codificador arrojó la imagen codificada; que al ser comparada con la imagen codificada de nuestro diseño produjo una imagen idéntica. Lo que se puede interpretar como un correcto funcionamiento en los bloques de software que se diseñaron.



a) b)

Figura 4-14 Comprobación del funcionamiento del codificador, a) imagen codificada con el diseño Reed-Solomon (7,3), b) imagen codificada con la función de MATLAB

En la Figura 4.14 se puede observar los resultados de la comprobación de los codificadores Reed-Solomon. En estos resultados se puede observar de manera visual que entre las dos figuras mostradas no hay diferencias. Lo cual refleja el buen funcionamiento de los codificadores como lo había explicado con anterioridad.

Ya habiendo obtenido los resultados de la prueba del algoritmo de codificación y que éstos comprueban su correcto funcionamiento, y me dispongo a implementarlo en el lenguaje de descripción de hardware VHDL, el cuál realiza la configuración de dispositivos de lógica programable tales como el FPGA.

4.4. Comprobación del funcionamiento de los bloques en XILINX ISE 10.1.

4.4.1. Obtención de la información a codificar.

A diferencia de la simulación en MATLAB, en la programación de XILINX ISE 10.1 la obtención de información fue de diferente forma, ya que se solicitó en el programa que se dieran 3 series de 3 símbolos cada serie de entrada; y éstos, a su vez, está conformado por 3 bits como lo muestra la Figura 4.15.

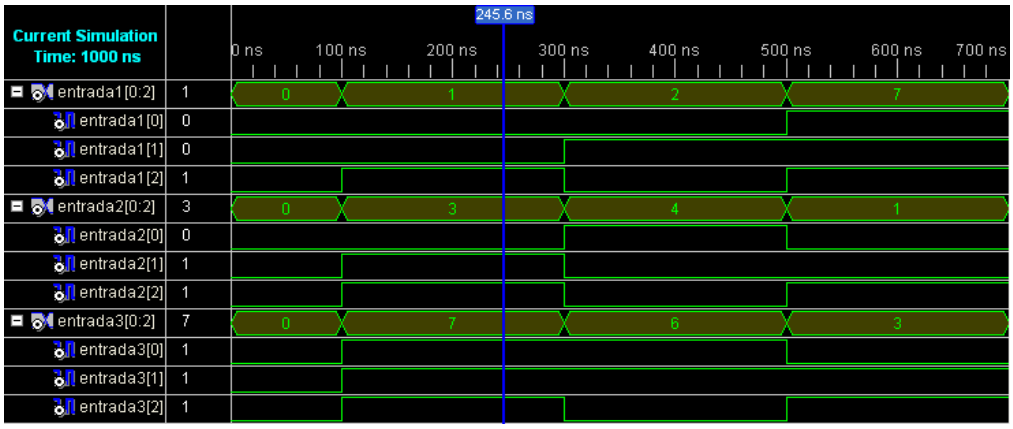


Figura 4-15 Información de entrada de 3 series de 3 símbolos cada uno

Posteriormente ésta información la ingrese al bloque de software, el cual es el mismo que el utilizado en la simulación de MATLAB pero ahora en VHDL; es decir seguí el mismo patrón que denota el diagrama de flujo de la Figura 4.10, pero con sus normas y especificaciones del lenguaje descriptivo de hardware.

Los primeros 3 bits del primer símbolo que entra al codificador es el denotado en la figura como **entrada1[0]**, posteriormente el que le sigue es **entrada1[1]** y así de forma sucesiva, hasta llegar a la **entrada3 [3]**.

Para demostrar de forma esquemática lo realizado en el programa XILINX ISE 10.1 tomé como ejemplo la programación del codificador Reed-Solomon de relación de codificación RS (7, 3). Por lo que por cada 3 símbolos que entre al bloque de software que diseñe para este tipo de codificador, se producirá 7 símbolos de palabra código.

4.4.2. Resultados.

En la Figura 4.16, se puede percibir tanto las entradas que se ingresaron al bloque del codificador Reed-Solomon como las salidas del mismo.

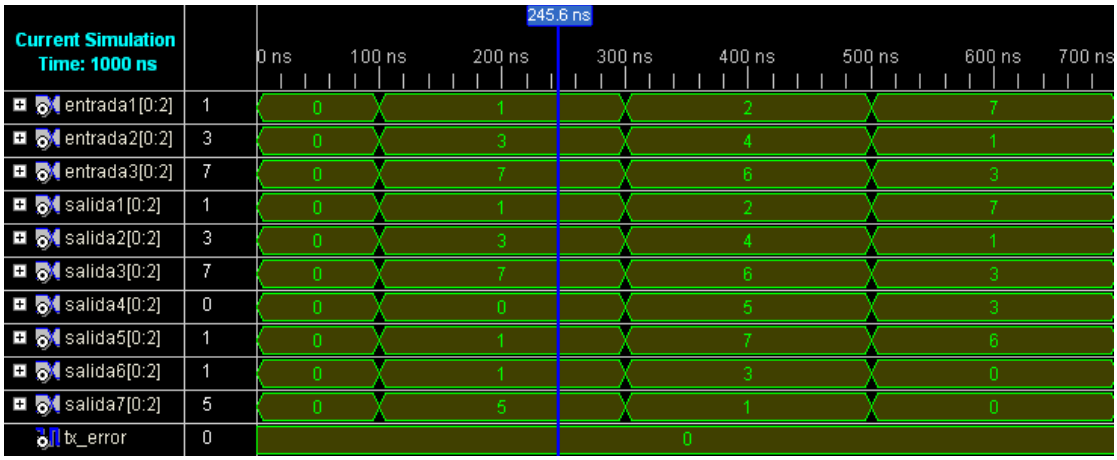


Figura 4-16 Entradas y Salidas del codificador de RS (7, 3)

Mediante ésta figura se puede observar que están entrando 3 series de 3 símbolos al codificador y que están saliendo 3 series de 7 símbolos, con un error de 0, por lo que la relación de codificación se cumple.

El software XILINX ISE 10.1 al momento de simular el codificador genera el diagrama del circuito que se implantará en la FPGA. En el caso del codificador RS (7, 3) el circuito que genera es el denotado en la Figura 4.17.

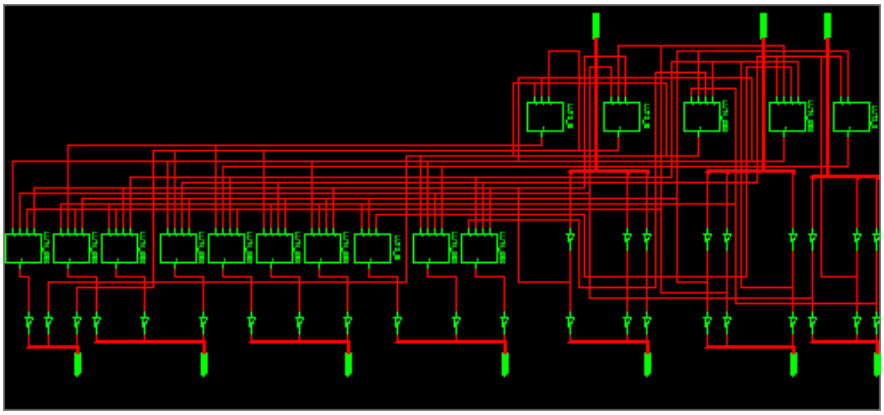
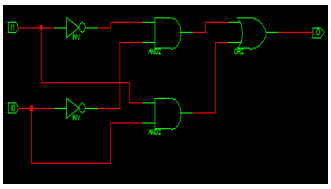
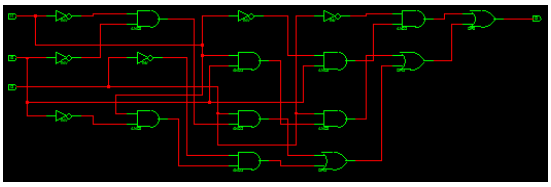


Figura 4-17 Esquema del circuito originado en la simulación para el codificador RS (7, 3)

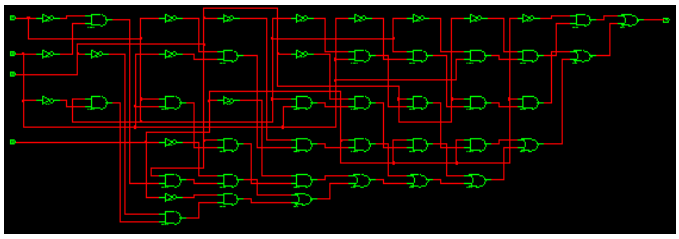
Éste circuito es un arreglo de bloques que en su interior tienen un arreglo de compuertas; aplicando un zoom a los bloques tendremos las siguientes compuertas mostradas en la siguiente figura.



a)



b)



c)

Figura 4-18 Diferentes arreglos de compuertas utilizadas en el diseño del circuito del codificador

El proceso de implementación de la FPGA que utilice en el codificador Reed-Solomon se visualiza en la Figura 4.19. Dicho proceso, se inicia en la especificación del objetivo, posteriormente se programa en lenguaje descriptivo de hardware, de la misma forma se comprueban los resultados de nuestro código en VHDL.

Obteniendo resultados positivos en el programa, se revisa la síntesis del esquema y se implementa la programación que es un esquema virtual; para finalizar se realiza la grabación de la FPGA para probar su funcionamiento.

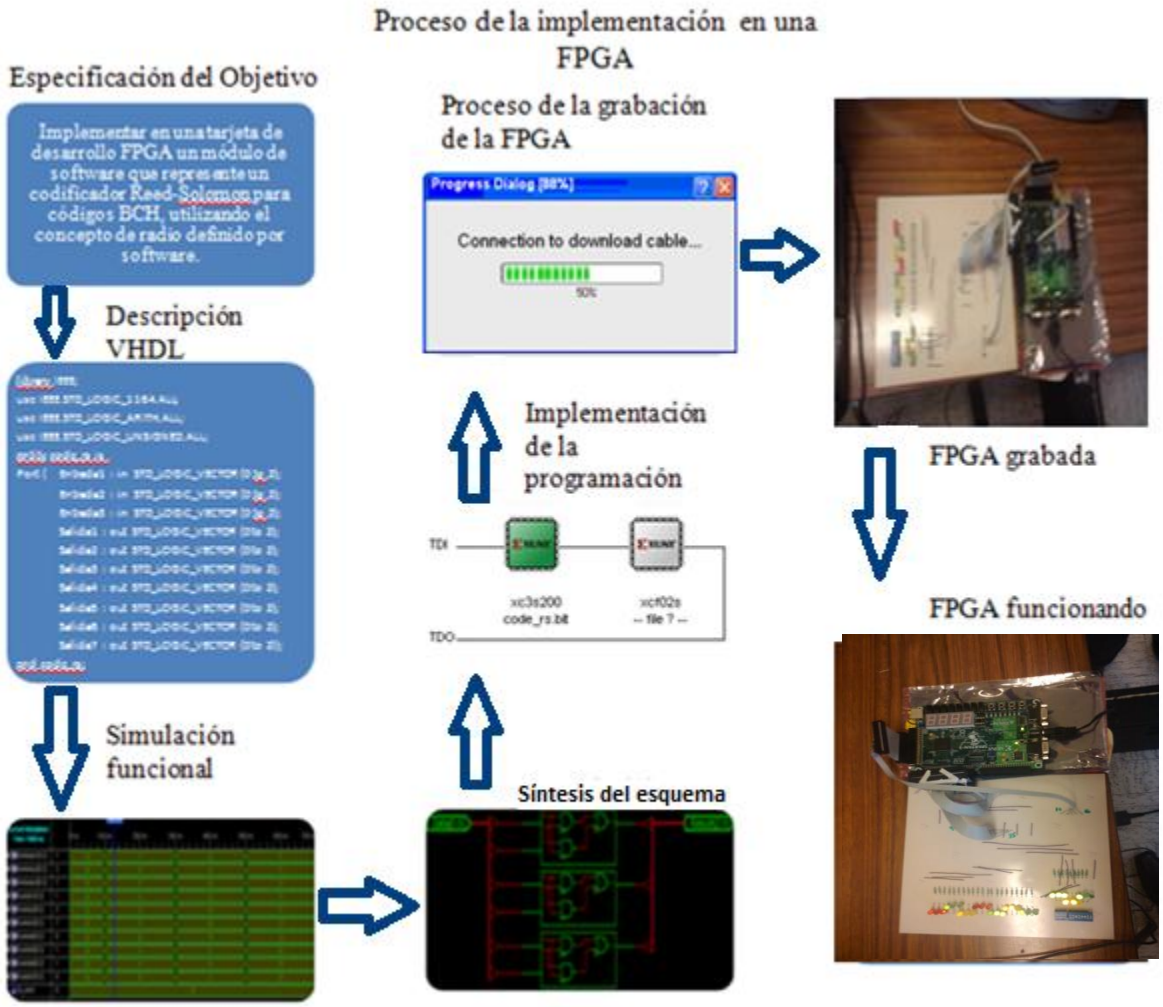


Figura 4-19 Proceso de implementación de una FPGA

Estos bloques de software obtenidos mediante la programación en VHDL con la herramienta de XILINX 10.1 me permite configurar las FPGA, de éste modo se obtienen diferentes codificadores Reed-Solomon en un solo dispositivo (hardware), al cual, con la simple actualización o cambio del software podrá funcionar como cualquiera de los codificadores Reed-Solomon utilizados en la industria.

Aunque en la actualidad estándares como DVB-S o DVB-T necesitan codificadores Reed-Solomon en hardware (circuitos), con éste proyecto demuestro que dichos circuitos también pueden implementarse a nivel de software de manera igualmente efectiva, por lo que se puede decir que las funciones proporcionadas por el hardware pueden ser realizadas por bloques de software de manera exitosa.

CONCLUSIONES

La presente investigación tuvo como objetivo la implementación en una tarjeta de desarrollo FPGA un módulo de software que represente un Codificador Reed-Solomon para códigos BCH utilizando el concepto de radio definido por software.

Para obtener esto, demostré que el Radio Definido por Software tiene una característica única en comparación con otros tipos de radios; ya que posee la capacidad de ser transformado a través del uso de software o de lógica reprogramable, esto se hace mediante programas desde un ordenador que posteriormente son introducidos a las FPGA's de propósito general utilizando un lenguaje descriptivo de hardware, puesto que es el lenguaje con características de modelado y arquitectura con amplia variedad de diseño.

Por lo anterior, procedí al análisis de los estándares DVB considerada como la organización encargada de la comunicación digital. Cabe mencionar que esta organización maneja el concepto de HDTV siendo este apreciado como uno de los principales medios de comunicación, ya que su estructura está conformada por un Codificador Reed-Solomon (204,188).

Debido a la gran importancia de este tema me vi en la necesidad de investigar el algoritmo, así como el funcionamiento del Codificador RS desde sus principios por los códigos BCH que son generados por los Códigos cíclicos de redundancia; es decir, a la palabra del mensaje se le agregan BITS de paridad formando la palabra código de longitud *k*, tal y como se detalla en el Capítulo 2; obteniendo de tal forma la programación del Codificador Reed-Solomon (7,3).

Básicamente, desarrolle la metodología para implementar un Codificador Reed-Solomon en una FPGA para cualquier especificación de detección y corrección de errores en aplicaciones del estándar DVB (DVB-C, DVB-S y DVB-T).

Por lo tanto, comprobé que el código Reed-Solomon es uno de los elementos importantes empleados en la digitalización del video y audio; así como también, en la aplicación de comunicaciones móviles y dispositivos de almacenamiento.

Respecto de las aplicaciones, el codificador RS es primordial de modo que, la vanguardia ha obligado a la tecnología a seguir avanzando tanto en las comunicaciones móviles como en las redes inalámbricas, haciendo más rápido el envío y recepción de datos, esto se refleja en las generaciones de las comunicaciones.

Los resultados son muy relevantes, ya que obtuve un módulo de software que representa a un codificador RS y que puede ser utilizado en el transmisor de cualquier categoría del estándar DVB, cabe resaltar que la metodología puede emplearse para implementar el Codificador RS (204,188) como se revisó en la Figura 1.3.

Se realizó un buen análisis, la FPGA fue programada con un lenguaje descriptivo de hardware (VHDL), que puede ser redefinido o sujeto a cambios con respecto a las especificaciones que se requieren con respecto al DVB estándar. La programación del lenguaje descriptivo de hardware fue desarrollada conforme a los lineamientos de investigación y análisis del objetivo de este estudio.

Para finalizar se sabe que este procedimiento se hace de modo mas simple por medio de la licencia que brinda la herramienta del ISE (Integrated Software Environment, Entorno Integrado de Software) de Xilinx por medio del System Generator (Simulink) aunque el inconveniente de este es el alto precio del codificador RS versión 5 lo que me permitió realizar un procedimiento más complejo pero minimizando costos y maximizando resultados, alcanzando la mejora para los usuarios y proveedores de comunicaciones digitales tal como el DVB, de modo que, puede utilizarse para cualquier especificación del codificador Reed-Solomon.

RECOMENDACIONES PARA TRABAJOS FUTUROS

En esta área se recomienda lo siguiente:

- Continuar estudios del Decodificador Reed-Solomon para completar el circuito de transmisión y recepción de datos.
- Además de realizar el diseño del decodificador Reed-Solomon para ser programado en una FPGA por medio del radio definido por software y consumar la programación del código con la funcionalidad de ambos bloques, codificador como decodificador.
- De igual forma, considerar la implementación en una tarjeta de desarrollo FPGA los elementos restantes que componen el transmisor DVB para culminar el esquema, estos son: la codificación de la fuente, el entrelazado, el codificador convolucional, el filtrado y las diferentes modulaciones.
- Otra recomendación es estudiar e implementar el receptor DVB para concluir el esquema transceptor.

ANEXOS

ANEXO A: INSTALACION DE SOFTWARE

Para realizar esta tesis se debe tener el siguiente software instalado antes de ejecutar el System Generator.

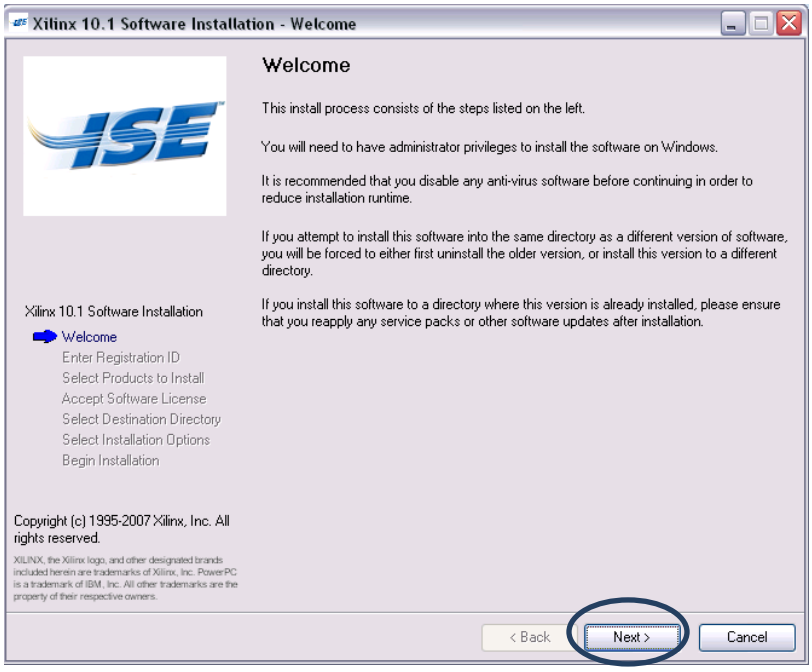
- Una de las siguientes versiones de MATLAB de The MathWorks Inc.:
  - MATLAB v7.4/Simulink v6.6 (R2007a)
  - MATLAB v7.5/Simulink v7.0 (R2007b)
- Xilinx ISE versión 10.1

Una característica de System Generator es, que requiere del siguiente software para ser instalado:

- Una herramienta de síntesis lógica. System Generator es totalmente compatible con Xilinx XST (incluido en el paquete ISE Foundation) y Synplify Pro v8.6.2 o v8.9 de Synplicity, Inc.
- Un hardware de descripción de lenguaje (HDL) simulador sólo para la simulación de los módulos de HDL en Simulink usando System Generator. System Generator interfaces de cosimulacion HDL son compatibles con el simulador de Xilinx ISE, ModelSim Xilinx Edición MXE (una opción con la Fundación ISE), y ModelSim PE o SE, versión v6.3c, del modelo Technology Inc.

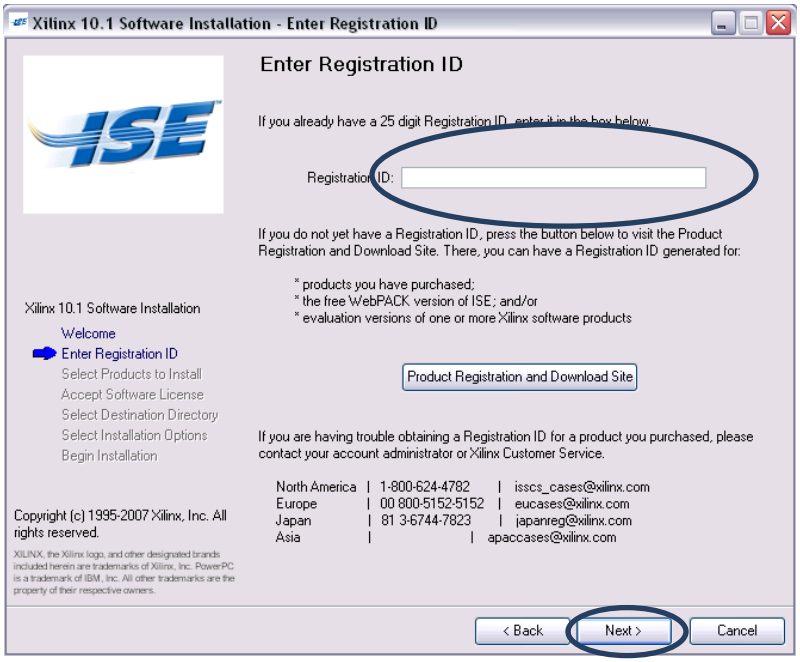
Pasos de instalación de ISE de Xilinx.

Este es el asistente de instalación de Xilinx 10.1 para la instalación del ISE.



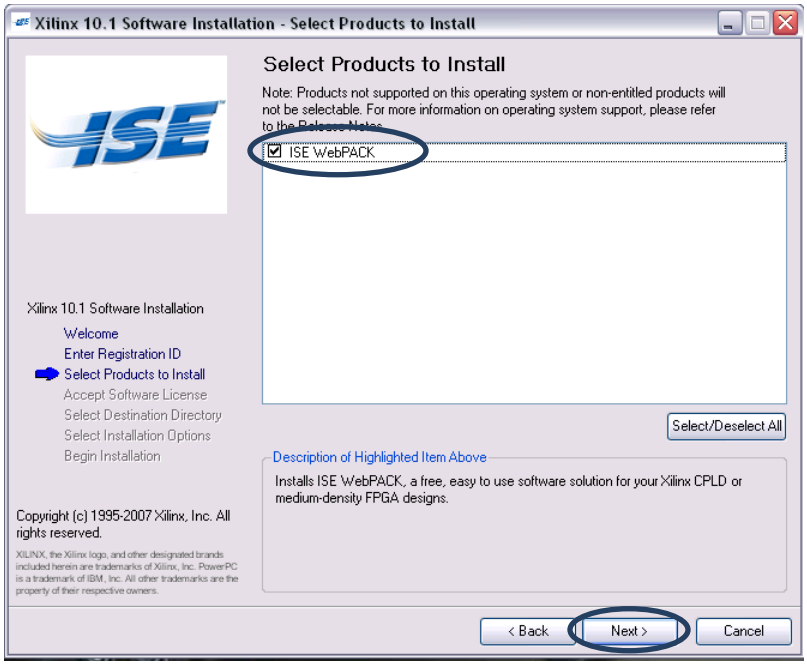
De click en siguiente para empezar con la instalación.

Le pedirá un registro ID; este registro es proporcionado por Xilinx, después de crear una cuenta en la página de Xilinx. Una vez introducida la clave de click en siguiente.

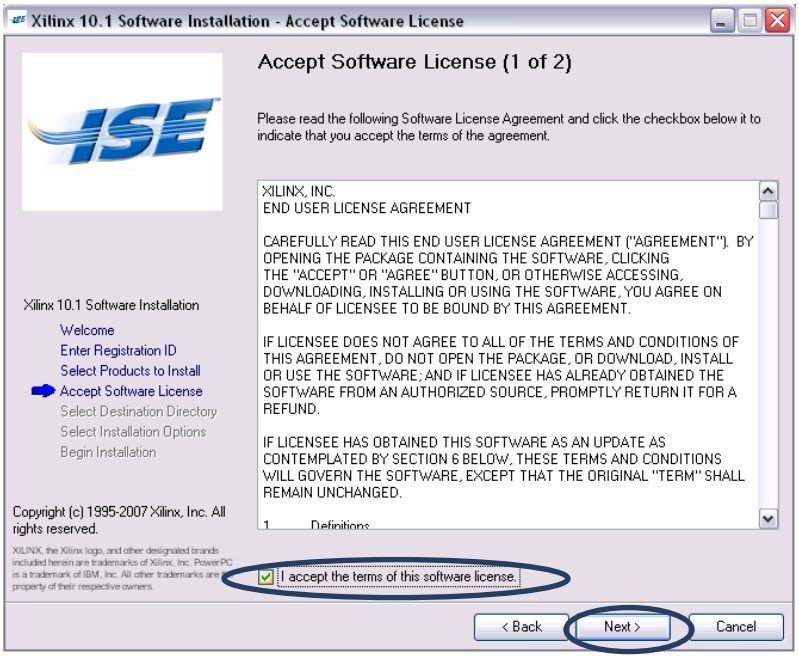


NOTA: no todos los registros funcionan con el System Generator, tendrá que conseguir un registro para su funcionamiento adecuado.

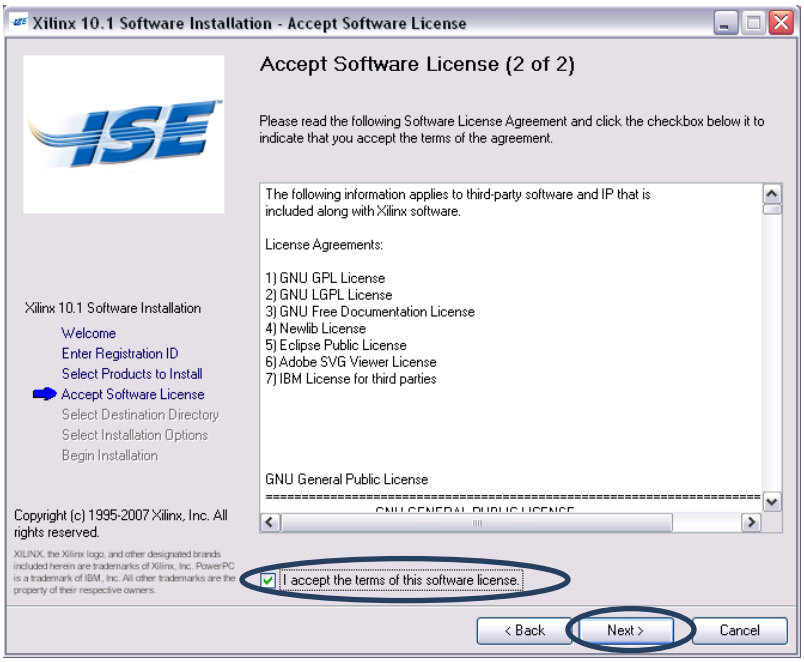
Seleccione la casilla de ISE WEBPACK si es que no está seleccionada, si la casilla no está seleccionada o activa, entonces su registro no es apto para ISE. Después de la selección de click en siguiente.



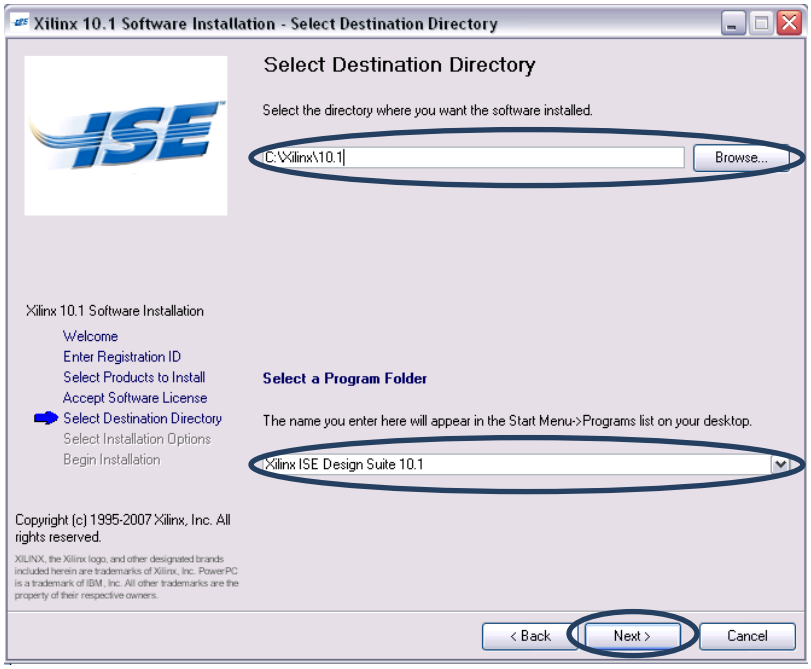
Acepte la primera licencia de Xilinx, habilitando la casilla de “Acepto los términos de la licencia” ubicada en la parte inferior. Y de click en siguiente.



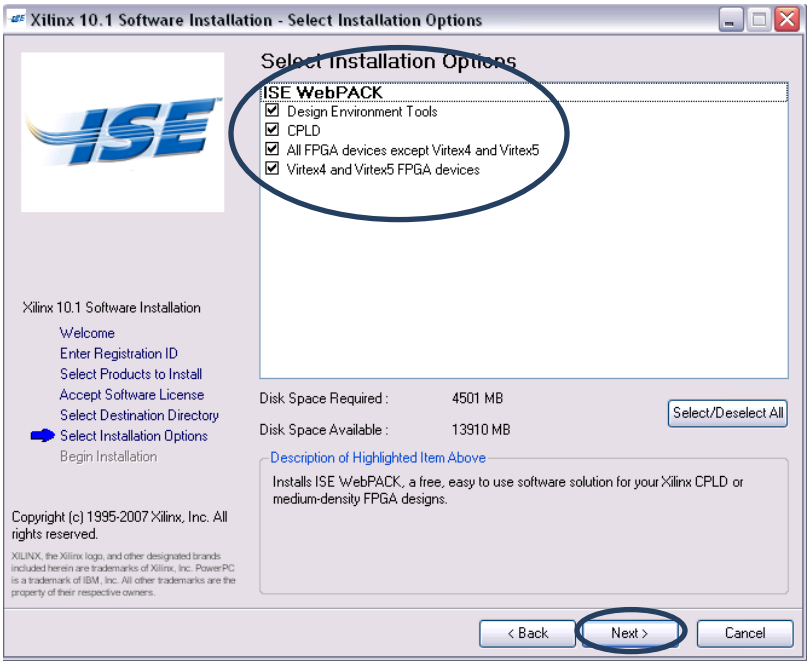
Acepte la segunda licencia que es para el público en General. Nuevamente habilite la casilla de aceptación de los términos de licencia. Y de click en siguiente.



Especifique la ruta donde se instalará Xilinx 10.1 y el nombre que aparecerá en el menú inicio. Después de seleccionar la ruta de click en siguiente.



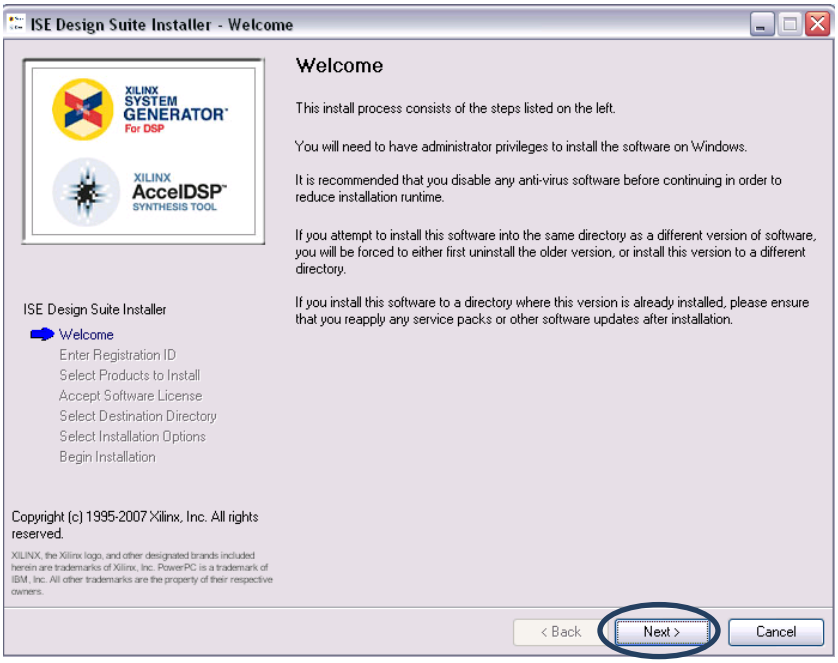
Seleccione los componentes opcionales de la instalación, marcando la casilla deseada se recomienda marcar todas.



Posteriormente de click en siguiente; mostrara el proceso de instalación de los componentes. Antes de terminar dá la opción de descargar las actualizaciones; como recomendación descárguelas todas. Terminando la actualización de click en finalizar.

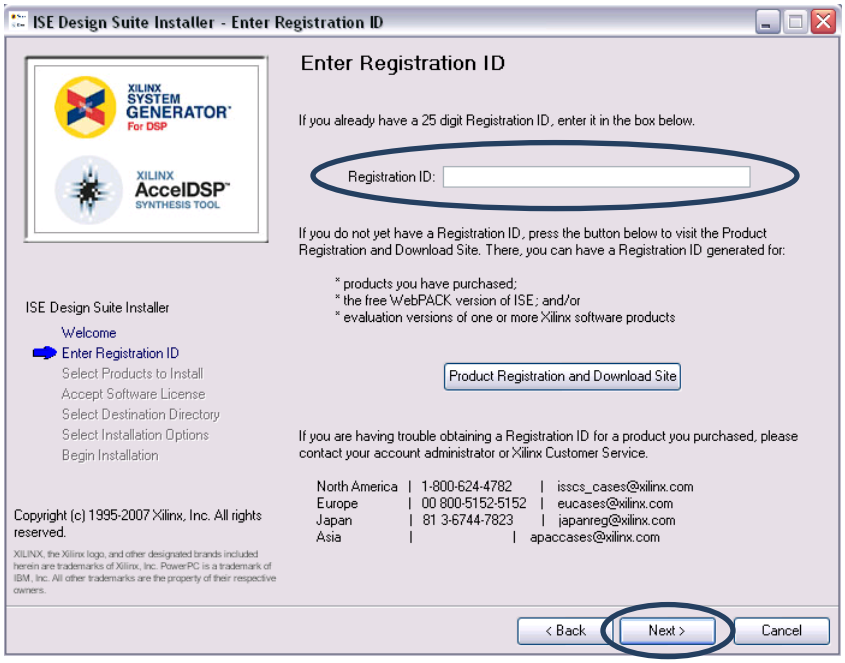
Pasos de instalación de System Generator de Xilinx.

Este es el asistente de instalación de Xilinx 10.1 para System Generator.



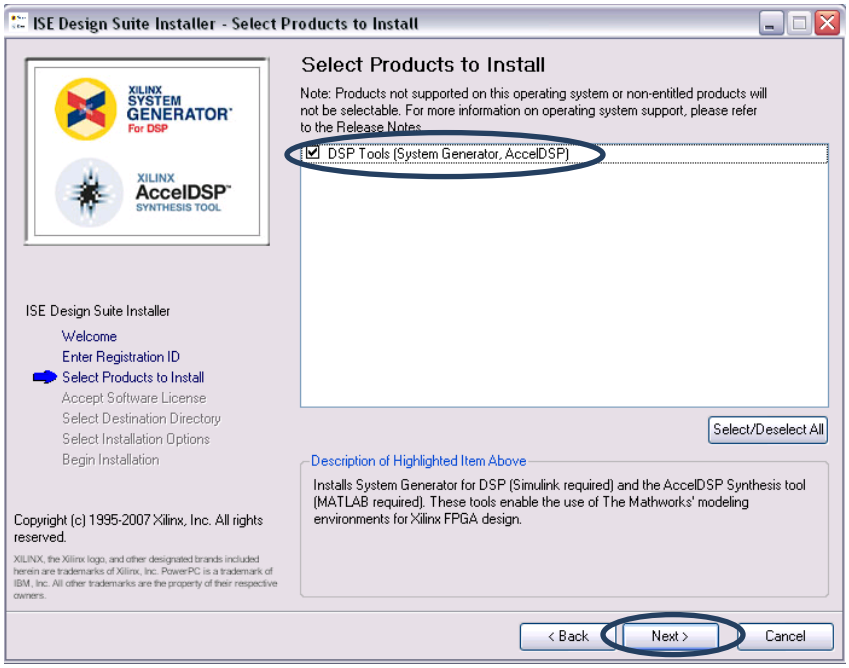
De click en siguiente para continuar.

Le pedirá un registro ID; este registro es proporcionado por Xilinx, después de crear una cuenta en la página de Xilinx. Después de click en siguiente.

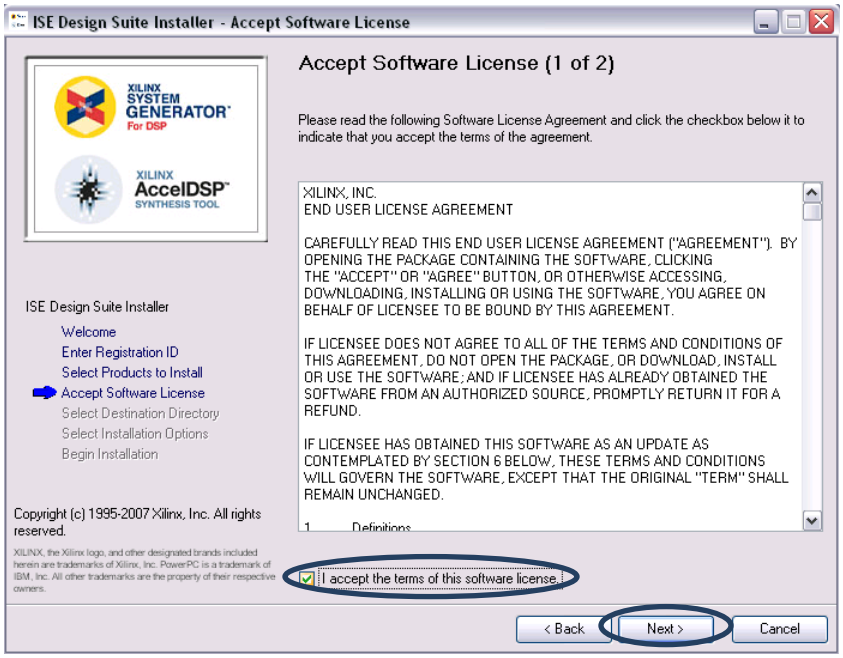


NOTA: el registro ya no esta disponible en Xilinx, pero puede buscarlo en alguna pagina de internet.

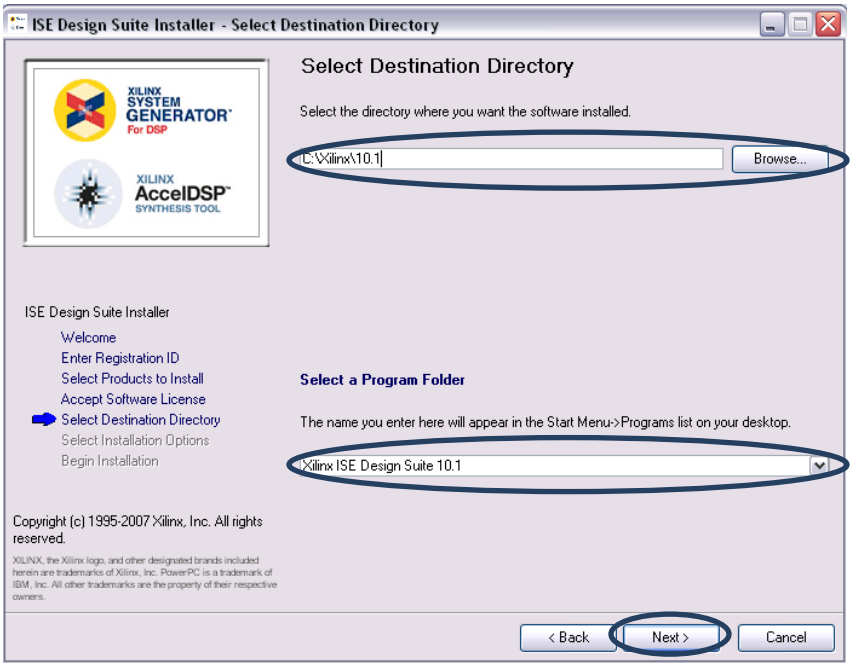
Seleccione la casilla de DSP Tools, si la casilla no está seleccionada o activa su registro no es el adecuado para System Generator. A continuación de click en siguiente.



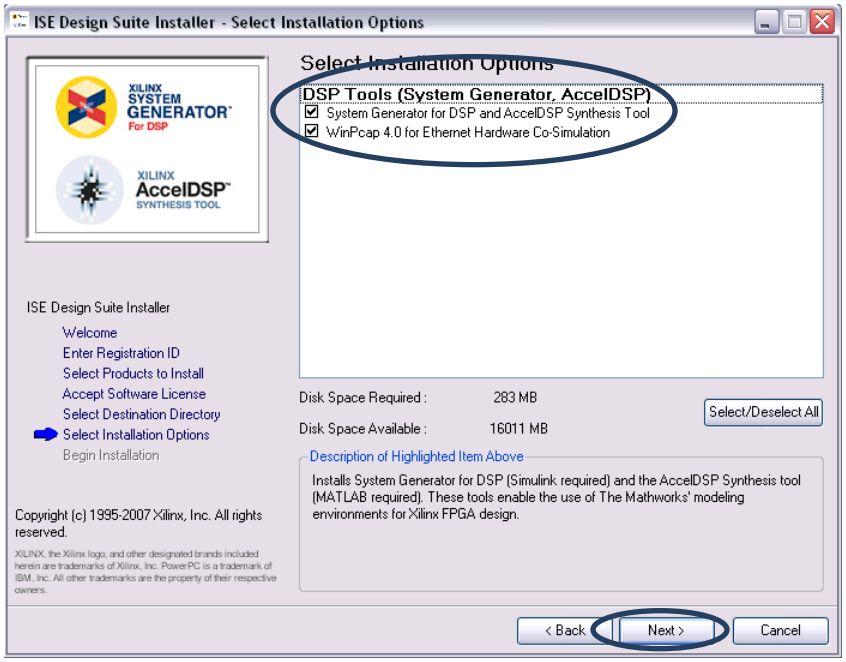
Acepte las dos licencias, una de Xilinx y una para el público en general, seleccionando la casilla de “aceptar los términos de la licencia”. Y de click en siguiente.



Especifique la ruta donde se instalará System Generator y el nombre con el cual aparecerá en el menú inicio. Después de click en siguiente.



Selecciones los componentes opcionales de la instalación, marcando la casilla deseada se recomienda marcar todas. Posteriormente de click en siguiente.



Antes de terminar dá la opción de descargar las actualizaciones; como recomendación descárguelas todas. Terminando la actualización de click en finalizar.

ANEXO B: PROGRAMACION EN C++

La programación en C++ para la comprobación de resultados del Codificador Reed-Solomon (7,3) es la siguiente.

```
#include <iostream.h>
#include <math.h>
#include <conio.h>

using namespace std;

void LeerEntrada (int Entrada [3])
{
    int i;
    cout <<endl<<endl<<"Introduce los simbolos a codificar: "<<endl<<endl;
    for (i=0; i<3; i++)
    {
       cout<<"Simbolo "<<i+1<<": ";
       cin>>Entrada [i];
    }
}

void ImprimeEntrada (int Entrada [3])
{
    cout<<endl<<"El mensaje introducido es: ";
    for (int i=0; i<3; i++)
    {
        cout<<Entrada[i];
        if (i<=1)
        {
              cout<<",";
        }
    }
}

int Galois(int NG, int Dd)
{
   int RG;

   switch(NG)
   {
          case 4:
              switch (Dd)
              {
                    case 0:
                        RG=0;
                        break;

                    case 1:
                        RG=4;                                          break;
```

```
            case 2:
                RG=3;
                break;

            case 3:
                RG=7;
                break;

            case 4:
                RG=6;
                break;

            case 5:
                RG=2;
                break;

            case 6:
                RG=5;
                break;

            case 7:
                RG=1;
                break;
        }
        break;

    case 5:
        switch (Dd)
        {
            case 0:
                RG=0;
                break;

            case 1:
                RG=5;
                break;

            case 2:
                RG=1;
                break;

            case 3:
                RG=4;
                break;

            case 4:
                RG=2;
                break;
```

Continuación del CASE

```
            case 5:
                RG=7;
                break;

            case 6:
                RG=3;
                break;

            case 7:
                RG=6;
                break;
        }
        break;

    case 7:
        switch (Dd)
        {
            case 0:
                RG=0;
                break;

            case 1:
                RG=7;
                break;

            case 2:
                RG=5;
                break;

            case 3:
                RG=2;
                break;

            case 4:
                RG=1;
                break;

            case 5:
                RG=6;
                break;

            case 6:
                RG=4;
                break;

            case 7:
                RG=3;
                break;
        }
```

```
                break;

    }
    return RG;
}

void DecBin (int Da[4],bool DatosBin [4][3])
{
    int i,j,k,l;

    for(i=0;i<4;i++)
    {
        for(j=0;j<3;j++)
        {
            DatosBin[i][j]=Da[i]%2;
            Da[i]=Da[i]/2;
        }
    }
    cout<<endl<<"Numero en binario: ";
    for(k=3;k>=0;k--)
    {
        for(l=2;l>=0;l--)
        {
            cout<<DatosBin[k][l];
        }
        if (k>=1)
        {
            cout<<",";
        }
    }
    cout<<endl;
}

int BinDec (bool DatoBin[4][3])
{
    int Dd,a,temp;
    Dd=0;
    cout<<endl;
    for(int l=0;l<3;l++)
    {
        //cout<<DatoBin[0][l];
        switch(DatoBin[0][l])
        {
            case 0:
                a=0;
                break;

            case 1:
                a=1;
                break;
```

```
                    default:
                        cout<<"Los digitos no son correctos"<<endl;
            }
            temp = a*pow (2,l);
            Dd = temp + Dd;
            //Decimal=Numbinario[a]* pow(2,a)+Decimal;*/
    }
    cout<<"En decimal es "<< Dd<<endl;
    return Dd;
}


void BinDec4 (bool DatoBin1[4][3],int CodDec[4])
{
    int a,temp;
    for(int k=3;k>=0;k--)
   {
            CodDec[k]=0;
   }

    //cout<<"El codigo es: ";
    temp=0;
    for(int l=0;l<3;l++)
    {
            //cout<<DatoBin1[0][l]<<" ";
            switch(DatoBin1[0][l])
            {
                case 0:
                    a=0;
                    break;

                case 1:
                    a=1;
                    break;

                default:
                     cout<<"Los digitos no son correctos"<<endl;
             }
             temp = a * pow (2,l);
             CodDec[0] = temp + CodDec[0];
             //Decimal=Numbinario[a]* pow(2,a)+Decimal;*/
    }
    //cout<<CodDec[0]<<", ";

    temp=0;
    for(int l=0;l<3;l++)
    {
            //cout<<DatoBin1[0][l]<<" ";
            switch(DatoBin1[1][l])
            {
```

```
            case 0:
                a=0;
                break;

            case 1:
                a=1;
                break;

            default:
                 cout<<"Los digitos no son correctos"<<endl;
        }
        temp = a * pow (2,l);
        CodDec[1] = temp + CodDec[1];
        //Decimal=Numbinario[a]* pow(2,a)+Decimal;*/
}
//cout<<CodDec[1]<<", ";

temp=0;
for(int l=0;l<3;l++)
{
        //cout<<DatoBin1[0][l]<<" ";
        switch(DatoBin1[2][l])
        {
            case 0:
                a=0;
                break;

            case 1:
                a=1;
                break;

            default:
                 cout<<"Los digitos no son correctos"<<endl;
        }
        temp = a * pow (2,l);
        CodDec[2] = temp + CodDec[2];
        //Decimal=Numbinario[a]* pow(2,a)+Decimal;*/
}
//cout<<CodDec[2]<<", ";

temp=0;
for(int l=0;l<3;l++)
{
        //cout<<DatoBin1[0][l]<<" ";
        switch(DatoBin1[3][l])
        {
            case 0:
                a=0;
                break;

            case 1:
```

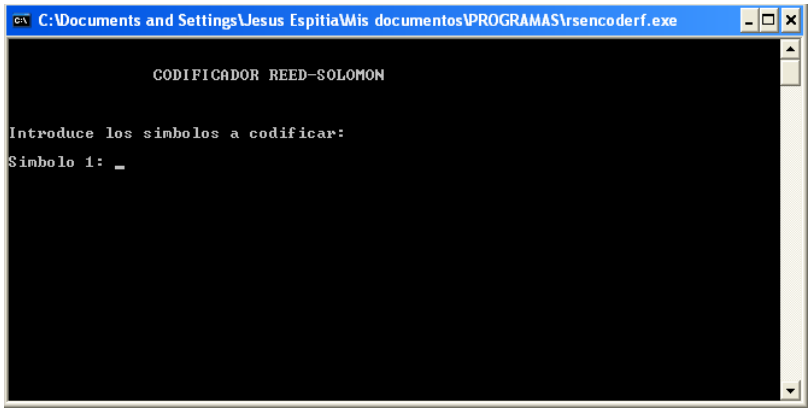
```
                    a=1;
                    break;

                default:
                     cout<<"Los digitos no son correctos"<<endl;
            }
            temp = a * pow (2,l);
            CodDec[3] = temp + CodDec[3];
            //Decimal=Numbinario[a]* pow(2,a)+Decimal;*/
    }
    //cout<<CodDec[3]<<", "<<endl;

}


void XorR4 (bool Operacion[4][3],bool Daxor[4][3],bool Resultado[4][3])
{
    cout<<endl<<"El resultado de la X-OR es: ";
    for(int l=2;l>=0;l--)
    {
        //cout<<Operacion[0][l];
        //cout<<Daxor[1][l];
        Resultado[0][l] = Operacion[0][l] xor Daxor[1][l];
        cout<<Resultado[0][l];
    }
    cout<<endl;
}

void XorR4_2 (bool Operacion[4][3],bool Daxor[4][3],bool Resultado[4][3])
{
    cout<<endl<<"El resultado de la X-OR es: ";
    for(int l=2;l>=0;l--)
    {
        //cout<<Operacion[0][l];
        //cout<<Daxor[1][l];
        Resultado[0][l] = Operacion[0][l] xor Daxor[2][l];
        cout<<Resultado[0][l];
    }
    cout<<endl;
}


void Xor (bool Operacion[4][3],bool Daxor[4][3],bool Resultado[4][3])
{
    bool CopiaBin[4][3];
    for(int k=3;k>=0;k--)
    {
        for(int l=2;l>=0;l--)
        {
            CopiaBin[k][l]= Operacion[k][l];
        }
```

```
    }

    for(int l=2;l>=0;l--)
    {
        Operacion[3][l]=Daxor[3][l];
    }

    for(int l=2;l>=0;l--)
    {
        Operacion[2][l]=Daxor[2][l] xor CopiaBin[3][l];
    }

    for(int l=2;l>=0;l--)
    {
        Operacion[1][l]=Daxor[1][l] xor CopiaBin[2][l];
    }

    for(int l=2;l>=0;l--)
    {
        Operacion[0][l]=Daxor[0][l] xor CopiaBin[1][l];
    }
}

int main(void)
{
   system("cls");
   int Entrada [3],Salida[7], Dd, Da[4],Codigo[4];
   bool
DaBin[4][3],DdBin[3],DaBin2[4][3],ResulDaBin[4][3],ResulDaBin1[4][3],Registro[4][3];
   char resp;
   do
   {
       system("cls");
       cout <<endl<<endl<< "              CODIFICADOR REED-SOLOMON" << endl << endl;
       LeerEntrada(Entrada);
       ImprimeEntrada(Entrada);
       Salida[0]=Entrada[0];
       Salida[1]=Entrada[1];
       Salida[2]=Entrada[2];

       cout<<endl<<"--------------PRIMERA VUELTA----------"<<endl;
       Dd=Entrada[0];
       Da[3]=Galois(5,Dd);
       Da[2]=Galois(7,Dd);
       Da[1]=Galois(7,Dd);
       Da[0]=Galois(4,Dd);

       cout << endl<<"El resultado de la funcion: " << Da[3]<<", "<<Da[2]
       <<", "<<Da[1]<<", "<<Da[0]<<endl;
       DecBin(Da,DaBin);
```

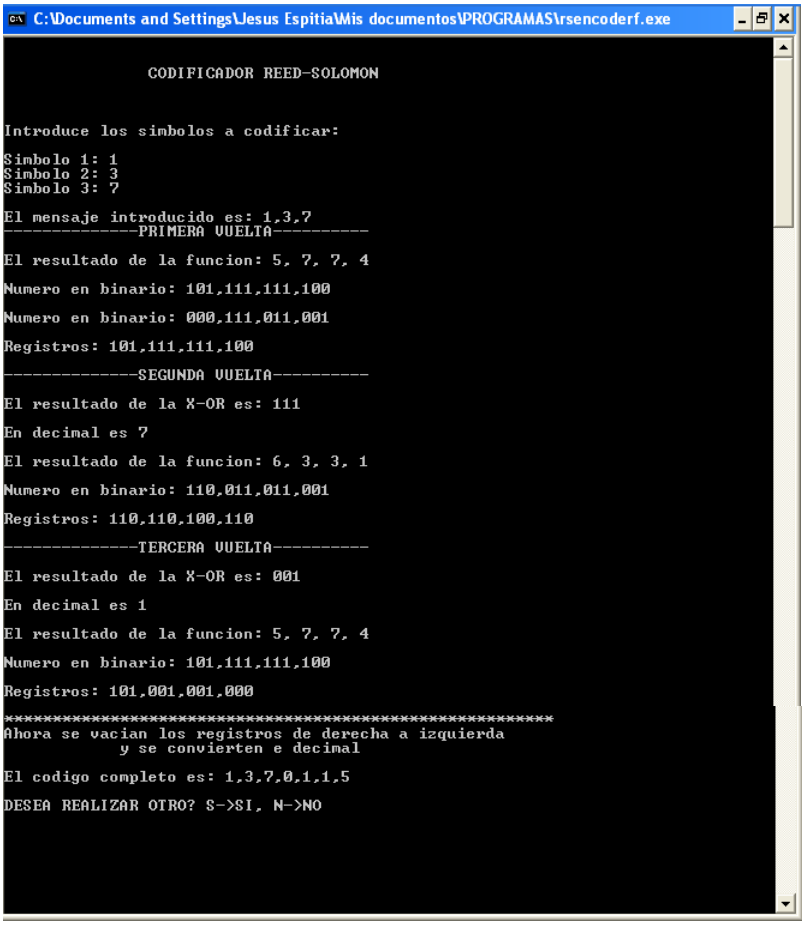
```
DecBin(Entrada,DaBin2);

for(int k=3;k>=0;k--)
{
        for(int l=2;l>=0;l--)
        {
           Registro[k][l]=DaBin[k][l];
        }
}
cout<<endl<<"Registros: ";
for(int k=3;k>=0;k--)
{
        for(int l=2;l>=0;l--)
        {
           cout<<Registro[k][l];
        }
        if (k>=1)
        {
          cout<<",";
        }
}
cout<<endl;

cout<<endl<<"--------------SEGUNDA VUELTA----------"<<endl;
XorR4(DaBin,DaBin2,ResulDaBin);
Dd=BinDec(ResulDaBin);
Da[3]=Galois(5,Dd);
Da[2]=Galois(7,Dd);
Da[1]=Galois(7,Dd);
Da[0]=Galois(4,Dd);

cout << endl<<"El resultado de la funcion: " << Da[3]<<", "<<Da[2]
<<", "<<Da[1]<<", "<<Da[0]<<endl;

DecBin(Da,DaBin);
Xor(Registro,DaBin,ResulDaBin);

cout<<endl<<"Registros: ";
for(int k=3;k>=0;k--)
{
        for(int l=2;l>=0;l--)
        {
          cout<<Registro[k][l];
        }
        if (k>=1)
        {
          cout<<",";
        }
}

cout<<endl<<endl<<"--------------TERCERA VUELTA----------"<<endl;
```

```
        XorR4_2(Registro,DaBin2,ResulDaBin);
        Dd=BinDec(ResulDaBin);
        Da[3]=Galois(5,Dd);
        Da[2]=Galois(7,Dd);
        Da[1]=Galois(7,Dd);
        Da[0]=Galois(4,Dd);
        cout << endl<<"El resultado de la funcion: " << Da[3]<<", "<<Da[2]
        <<", "<<Da[1]<<", "<<Da[0]<<endl;

        DecBin(Da,DaBin);
        Xor(Registro,DaBin,ResulDaBin);
        cout<<endl<<"Registros: ";
        for(int k=3;k>=0;k--)
        {
                for(int l=2;l>=0;l--)
                {
                  cout<<Registro[k][l];
                }
                if (k>=1)
                {
                  cout<<",";
                }
        }

        cout<<endl<<endl;
        cout<<"************************************************************";
        cout<<endl<<"Ahora se vacian los registros de derecha a izquierda";
        cout<<endl<<"            y se convierten e decimal";
        cout<<endl<<endl<<"El codigo completo es: ";
        BinDec4(Registro,Codigo);
        Salida[3]=Codigo[0];
        Salida[4]=Codigo[1];
        Salida[5]=Codigo[2];
        Salida[6]=Codigo[3];
        for(int i=0;i<7;i++)
        {
                cout<<Salida[i];
                if (i<=5)
                {
                   cout<<",";
                }
        }

        cout<<endl<<endl;

        cout<<"DESEA REALIZAR OTRO? S->SI, N->NO"<<endl;
        cin>>("%s",&resp);
        //system("pause");
    }while(resp=='s'||resp=='S');
}
```

Para así obtener la pantalla de captura



Para mostrar su funcionamiento, introducimos la palabra código 1, 3, 7 con palabra codificada 1, 3, 7, 0, 1, 1, 5.



Como pueden observar el programa realiza la codificación rápida y confiable, con la opción de realizar más de una codificación.

ANEXO C: PROGRAMACION EN MATLAB

Para obtener resultados más certeros, decidí introducir información proporcionada por una imagen realizando un programa en MATLAB, siendo el programa más complejo para la manipulación de imágenes y funciones matemáticas.

Primero introduje una imagen como la siguiente:

El programa siguiente es para obtener el codificador Reed-Solomon (7,3).

```
%% INSTITUTO POLITECNICO NACIONAL.
%% UNIDAD PROFESIONAL "ADOLFO LÓPEZ MATEOS", ZACATENCO
%% ESCUELA SUPERIOR DE INGENIERIA MECANICA Y ELECTRICA.
%% INGENIERIA EN COMUNICACIONES Y ELECTRONICA.
%% ELABORO: JESUS ESPITIA JUAREZ.
%% ASESOR: M. EN C. DAVID VAZQUEZ ALVAREZ.
%% ASESOR: M. EN C. GABRIELA SANCHEZ MELENDEZ.

%% PROGRAMA QUE SIMULA UNA CODIFICACION REED-SOLOMON (7,3).

clear all;%Limpia todas las ventanas
close all;%cierra todas las ventanas
clc;%Limpia la ventana de comandos

%%


%%
A=imread('Orca01','jpg');%Se lee la imagen
figure
imshow(A), title('IMAGEN ORIGINAL');%Muestra la imagen original
%A

nivel = graythresh(A);%Valor de umbral
bw = im2bw(A,nivel);%Imagen binarizada
figure
imshow(bw), title('IMAGEN BINARIZADA');
%bw
%Obtención del vector del mensaje.
[fil,col]=size(bw);
cont=1;
```

```
for i=1:fil
    for j=1:col
        B(cont)=bw(i,j);
        cont=cont+1;
    end
end

%Imprime la imagen del vector que será el que se ingresará al
codificador.
% figure, imshow(B);
% title('IMAGEN DEL VECTOR DEL MENSAJE ORIGINAL');

%Obtención del vector que se utilizará para la comprobación de la
%información del mensaje original y la información decodificada.
con=1;
for i=1:fil
        for j=1:col
            OP(con)=bw(i,j);
            con=con+1;
        end
end
%OP
%Imprime la imagen del vector que será el que se ingresará al
codificador.
% figure, imshow(OP);
% title('IMAGEN DE LA COPIA DEL VECTOR DEL MENSAJE ORIGINAL');

%Obtencion de la matriz n*3
[fila,columna]=size(B);
vec=columna/3;
int c;
c=0;
for i=1:vec
    for j=1:3
        M(i,j)=B(1,j+c);
    end
    c=c+3;
end

%Imprime la imagen de la matriz n*3 que será el que se ingresará al
codificador.
figure, imshow(M);
title('IMAGEN DE LA MATRIZ N*3');
% disp('MATRIZ M:');
% M

%CONVERSION DE BINARIO A DECIMAL
%Los valores se convierten a strings por la funcion no acepta
%enteros.
[Bin1,Bin2]=size(M);

for i=1:Bin1
    for j=1:Bin2
        if(M(i,j)==1)
            S(i,j)='1';
        else
```

```
            S(i,j)='0';
        end
    end
end
% disp('MATRIZ BINARIA STRING');
% S

%Ahora se utiliza la funcion para obtener los numeros
%en decimal
D=bin2dec(S);
D(3843,1)=0;
% disp('MATRIZ DECIMAL');
% D

%Organizacion de los numeros en decimal para acomodarlos
%de 3 en 3.

[F,C]=size(D);
temp=F/3;

int cont;
cont=0;
for i=1:temp
    for j=1:3
            P(i,j)=D(i+cont,1);
            cont=cont+1;
    end
    cont=cont-1;
end
disp('MATRIZ DE ENTEROS');
P

%%
%% Aplicacion del Codificador Reed-Solomon (7,3)
n=7;%tamaño del mensaje
k=3;%numero de simbolos de dato
m=3;%numero de bits por simbolo
b=0;%corresponde al valor de i

genpoly=rsgenpoly(n,k,11,b);

% El cual permite obtener la palabra codificada
% compuesta por 3 símbolos de datos y 4 símbolos
% de paridad.
[Fi,Co]=size(P);

for i=1:Fi
    Dd=P(i,1);
    Registros(1,1)=0;
    Registros(1,2)=0;
    Registros(1,3)=0;
    Registros(1,4)=0;

    Da(1,4) = Galois (4 ,Dd);
    Da(1,3) = Galois (7 ,Dd);
```

```
    Da(1,2) = Galois (7 ,Dd);
    Da(1,1) = Galois (5 ,Dd);
    % disp('Campo de galois primera vuelta');
    % Da

    Registros=Da;
    % disp('Datos almacanedos');
    % Registros

    DaBin=dec2bin(Registros(1,4),3);

    %CONVERSION DE DECIMAL A BINARIO
    %Los valores se convierten a strings por la funcion no acepta
    %enteros.
    [Df,Dc]=size(DaBin);

    for y=1:Dc
        if(DaBin(1,y)=='1')
            DaB(1,y)=1;
        else
            DaB(1,y)=0;
        end
    end
    %DaB

    DaBin2=dec2bin(P(i,2),3);

    %CONVERSION DE DECIMAL A BINARIO
    %Los valores se convierten a strings por la funcion no acepta
    %enteros.
    [Df1,Dc1]=size(DaBin2);

    for y=1:Dc1
        if(DaBin2(1,y)=='1')
            DaB2(1,y)=1;
        else
            DaB2(1,y)=0;
        end
    end
    %DaB2
%%
%%
    XORR4 = xor (DaB, DaB2);
    %XORR4
    [X1,X2]=size(XORR4);
    for y=1:X2
        if(XORR4(1,y)==1)
            R4(1,y)='1';
        else
            R4(1,y)='0';
        end
    end
    %R4

    %% Campo de Galois segunda vuelta
```

```
    Dd=bin2dec(R4);
    Da(1,4) = Galois (4 ,Dd);
    Da(1,3) = Galois (7 ,Dd);
    Da(1,2) = Galois (7 ,Dd);
    Da(1,1) = Galois (5 ,Dd);
    %Da

    ReB=dec2bin(Registros,3);
    DaBi=dec2bin(Da,3);
    [Bina1,Bina2]=size(ReB);

    for x=1:Bina1
        for y=1:Bina2
            if(ReB(x,y)=='1')
                ReBin(x,y)=1;
            else
                ReBin(x,y)=0;
            end
        end
    end
    %ReBin

    [Binar1,Binar2]=size(DaBi);

    for x=1:Binar1
        for y=1:Binar2
            if(DaBi(x,y)=='1')
                ReBina(x,y)=1;
            else
                ReBina(x,y)=0;
            end
        end
    end
    %ReBina

    Registros=xor4(ReBin, ReBina);
%%
%%
    DaBin=dec2bin(Registros(1,4),3);

    %CONVERSION DE DECIMAL A BINARIO
    %Los valores se convierten a strings por la funcion no acepta
    %enteros.
    [Df,Dc]=size(DaBin);

    for y=1:Dc
        if(DaBin(1,y)=='1')
            DaB(1,y)=1;
        else
            DaB(1,y)=0;
        end
    end
    %DaB

    DaBin2=dec2bin(P(i,3),3);
```

```
    %CONVERSION DE DECIMAL A BINARIO
    %Los valores se convierten a strings por la funcion no acepta
    %enteros.
    [Df1,Dc1]=size(DaBin2);

    for y=1:Dc1
        if(DaBin2(1,y)=='1')
            DaB2(1,y)=1;
        else
            DaB2(1,y)=0;
        end
    end
    %DaB2
%%
%%
    XORR4 = xor (DaB, DaB2);
    %XORR4
    [X1,X2]=size(XORR4);
    for y=1:X2
        if(XORR4(1,y)==1)
            R4(1,y)='1';
        else
            R4(1,y)='0';
        end
    end
    %R4

    %% Campo de Galois tercera vuelta
    Dd=bin2dec(R4);
    Da(1,4) = Galois (4 ,Dd);
    Da(1,3) = Galois (7 ,Dd);
    Da(1,2) = Galois (7 ,Dd);
    Da(1,1) = Galois (5 ,Dd);
    %Da

    ReB=dec2bin(Registros,3);
    DaBi=dec2bin(Da,3);
    [Bina1,Bina2]=size(ReB);

    for x=1:Bina1
        for y=1:Bina2
            if(ReB(x,y)=='1')
                ReBin(x,y)=1;
            else
                ReBin(x,y)=0;
            end
        end
    end
    %ReBin

    [Binar1,Binar2]=size(DaBi);

    for x=1:Binar1
        for y=1:Binar2
            if(DaBi(x,y)=='1')
```

```
                ReBina(x,y)=1;
            else
                ReBina(x,y)=0;
            end
        end
    end
    %ReBina

    Registros=xor4(ReBin, ReBina);
%% Codigo completo (simbolos)
    Codigo(i,1)=P(i,1);
    Codigo(i,2)=P(i,2);
    Codigo(i,3)=P(i,3);
    Codigo(i,4)=Registros(1,4);
    Codigo(i,5)=Registros(1,3);
    Codigo(i,6)=Registros(1,2);
    Codigo(i,7)=Registros(1,1);
%     disp('CODIGO DE LA PRIMERA FILA');
%     Codigo
end
disp('CODIGO COMPLETO DE LA IMAGEN ORIGINAL');
Codigo

CodBin=dec2bin(Codigo,3);
%CodBin
[Codf,Codc]=size(CodBin);
ctafila=1;
ctacol=0;
for x=1:Codf
    for y=1:Codc
        if(y+ctacol==239)
            ctafila=ctafila+1;
            ctacol=0;
        end
        IC(ctafila,y+ctacol)=CodBin(x,y);
    end
    ctacol=ctacol+3;
end
%IC
[ICF,ICC]=size(IC);
for x=1:ICF
    for y=1:ICC
        if(IC(x,y)=='1')
            ICD(x,y)=1;
        else
            ICD(x,y)=0;
        end
    end
end
%%
ICD
figure
imshow(ICD), title('IMAGEN CODIFICADA');
%%
```

Esto se logro creando funciones para repetir procesos; fueron creados fuera del programa principal, puesto que así se trabaja en MATLAB.

Las funciones creadas utilizadas para la programación principal fueron las X-OR y los Campos de Galois.

Función X-OR

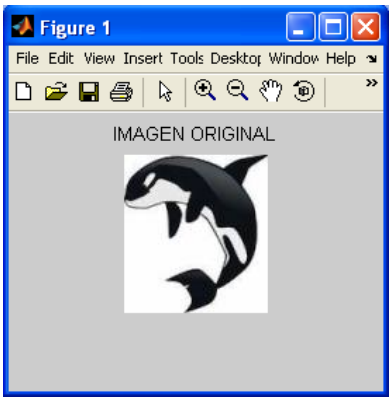
```
%% XOR DE LOS REGISTROS
function Registros = xor4 (ReBin, ReBina)
for i=1:3
    RegNBin(1,i)=ReBina(1,i);
end

for i=1:3
    RegNBin(2,i)=xor(ReBin(1,i), ReBina(2,i));
end

for i=1:3
    RegNBin(3,i)=xor(ReBin(2,i), ReBina(3,i));
end

for i=1:3
    RegNBin(4,i)=xor(ReBin(3,i), ReBina(4,i));
end

%RegNBin
[RNBF,RNBC]=size(RegNBin);

for i=1:RNBF
    for j=1:RNBC
        if(RegNBin(i,j)==1)
            RegNBinS(i,j)='1';
        else
            RegNBinS(i,j)='0';
        end
    end
end
Registros=bin2dec(RegNBinS);
Registros=Registros';
```

Función Campos de Galois

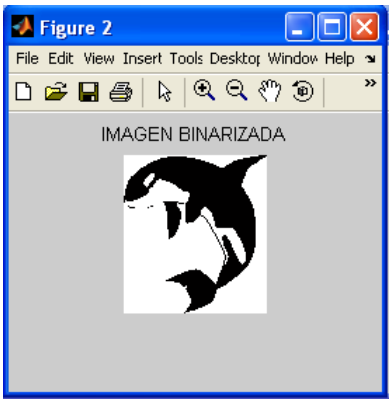
```
%Tabla de campo de Galois (4,5,7)
%Galois (NG, Dd) devuelve el valor del campo de galois
function RG = Galois (NG, Dd)
int RG;

switch(NG)
              case 4,
                   switch (Dd)
                          case 0,
                               RG=0;
```

```
                        case 1,
                            RG=4;
                        case 2,
                            RG=3;
                        case 3,
                            RG=7;
                        case 4,
                            RG=6;
                        case 5,
                            RG=2;
                        case 6,
                            RG=5;
                        case 7,
                            RG=1;
                    end

                case 5,
                    switch (Dd)
                        case 0,
                            RG=0;
                        case 1,
                            RG=5;
                        case 2,
                            RG=1;
                        case 3,
                            RG=4;
                        case 4,
                            RG=2;
                        case 5,
                            RG=7;
                        case 6,
                            RG=3;
                        case 7,
                            RG=6;
                    end

                case 7,
                    switch (Dd)
                        case 0,
                            RG=0;
                        case 1,
                            RG=7;
                        case 2,
                            RG=5;
                        case 3,
                            RG=2;
                        case 4,
                            RG=1;
                        case 5,
                            RG=6;
                        case 6,
                            RG=4;
                        case 7,
                            RG=3;
                    end
end
```

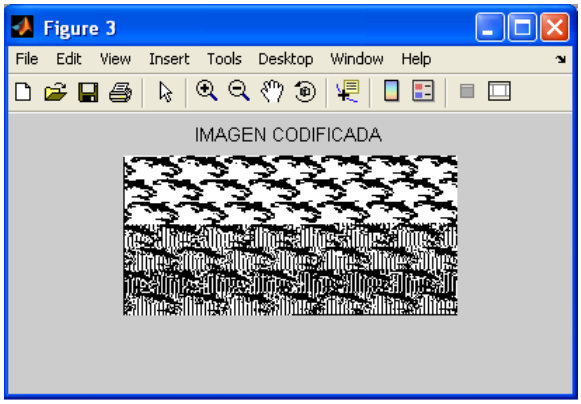
Y obtener el procedimiento con los resultados siguientes.



Después se binariza la imagen para poder manejar los bits y así formar símbolos de 3 bits.



Posteriormente, empieza la codificación de la imagen como se describió en el capitulo 2 de este trabajo; logrando la imagen consiguiente codificada.



Para poder comprobar mis resultados; realice el programa con funciones de MATLAB, siendo que la función ya está incluida, introduje la misma imagen para mostrar el procedimiento del Codificador Reed-Solomon (7, 3).

```
%% INSTITUTO POLITECNICO NACIONAL.
%% UNIDAD PROFESIONAL "ADOLFO LOPEZ MATEOS", ZACATENCO
%% ESCUELA SUPERIOR DE INGENIERIA MECANICA Y ELECTRICA.
%% INGENIERIA EN COMUNICACIONES Y ELECTRONICA.
%% ELABORO: JESUS ESPITIA JUAREZ.
%% ASESOR: M. EN C. DAVID VAZQUEZ ALVAREZ.
%% ASESOR: M. EN C. GABRIELA SANCHEZ MELENDEZ.

%% PROGRAMA QUE SIMULA UNA CODIFICACION REED-SOLOMON (7,3).

clear all;%Limpia todas las ventanas
close all;%cierra todas las ventanas
clc;%Limpia la ventana de comandos

%%


%%
A=imread('Orca01','jpg');%Se lee la imagen
% figure
% imshow(A), title('IMAGEN ORIGINAL');%Muestra la imagen original
%A

nivel = graythresh(A);%Valor de umbral
bw = im2bw(A,nivel);%Imagen binarizada
figure
imshow(bw), title('IMAGEN BINARIZADA');
% bw
%Obtención del vector del mensaje.
[fil,col]=size(bw);
cont=1;
for i=1:fil
    for j=1:col
        B(cont)=bw(i,j);
        cont=cont+1;
    end
end

%Imprime la imagen del vector que será el que se ingresará al
codificador.
% figure, imshow(B);
% title('IMAGEN DEL VECTOR DEL MENSAJE ORIGINAL');

%Obtención del vector que se utilizará para la comprobación de la
%información del mensaje original y la información decodificada.
con=1;
for i=1:fil
        for j=1:col
            OP(con)=bw(i,j);
            con=con+1;
        end
end
%OP
%Imprime la imagen del vector que será el que se ingresará al
codificador.
```

```
% figure, imshow(OP);
% title('IMAGEN DE LA COPIA DEL VECTOR DEL MENSAJE ORIGINAL');

%Obtencion de la matriz n*3
[fila,columna]=size(B);
vec=columna/3;
int c;
c=0;
for i=1:vec
    for j=1:3
        M(i,j)=B(1,j+c);
    end
    c=c+3;
end

%Imprime la imagen de la matriz n*3 que será el que se ingresará al
codificador.
% figure, imshow(M);
% title('IMAGEN DE LA MATRIZ N*3');
% disp('MATRIZ M:');
% M

%CONVERSION DE BINARIO A DECIMAL
%Los valores se convierten a strings por la funcion no acepta
%enteros.
[Bin1,Bin2]=size(M);

for i=1:Bin1
    for j=1:Bin2
        if(M(i,j)==1)
            S(i,j)='1';
        else
            S(i,j)='0';
        end
    end
end
% disp('MATRIZ BINARIA STRING');
% S

%Ahora se utiliza la funcion para obtener los numeros
%en decimal
D=bin2dec(S);
D(3843,1)=0;
% disp('MATRIZ DECIMAL');
% D

%Organizacion de los numeros en decimal para acomodarlos
%de 3 en 3.

[F,C]=size(D);
temp=F/3;

int cont;
cont=0;
for i=1:temp
```

```
    for j=1:3
            P(i,j)=D(i+cont,1);
            cont=cont+1;
    end
    cont=cont-1;
end
% disp('MATRIZ DE ENTEROS');
% P

%%
%% Aplicacion del Codificador Reed-Solomon (7,3)
n=7;%tamaño del mensaje
k=3;%numero de simbolos de dato
m=3;%numero de bits por simbolo
b=0;%corresponde al valor de i

genpoly=rsgenpoly(n,k,11,b);

% El cual permite obtener la palabra codificada
% compuesta por 3 símbolos de datos y 4 símbolos
% de paridad.
[Fi,Co]=size(P);

% funcion para formar los resultados de Campos de Galois para obtener los
% números decimales, insercion de 3 numeros decimales(pixel por pixel).
msg  =  gf([P(1,1)  P(1,2)  P(1,3);P(2,1)  P(2,2)  P(2,3);P(3,1)  P(3,2)
P(3,3);P(4,1)   P(4,2)   P(4,3);P(5,1)   P(5,2)   P(5,3);P(6,1)   P(6,2)
P(6,3);P(7,   1)   P(7,2)   P(7,3);P(8,1)   P(8,2)   P(8,3);P(9,1)   P(9,2)
P(9,3);P(10,1)  P(10,2)  P(10,3);P(11,1)  P(11,2)  P(11,3);P(12,1)  P(12,2)
P(12,3);P(13,1)  P(13,2)  P(13,3);P(14,1)  P(14,2)  P(14,3);P(15,1)  P(15,2)
P(15,3);P(16,1)  P(16,2)  P(16,3);P(17,1)  P(17,2)  P(17,3);P(18,1)  P(18,2)
P(18,3);P(19,1)  P(19,2)  P(19,3);P(20,1)  P(20,2)  P(20,3);P(21,1)  P(21,2)
P(21,3);P(22,1)  P(22,2)  P(22,3);P(23,1)  P(23,2)  P(23,3);P(24,1)  P(24,2)
P(24,3);P(25,1)  P(25,2)  P(25,3);P(26,1)  P(26,2)  P(26,3);P(27,1)  P(27,2)
P(27,3);P(28,1)  P(28,2)  P(28,3);P(29,1)  P(29,2)  P(29,3);P(30,1)  P(30,2)
P(30,3);P(31,1)  P(31,2)  P(31,3);P(32,1)  P(32,2)  P(32,3);P(33,1)  P(33,2)
P(33,3);P(34,1)  P(34,2)  P(34,3);P(35,1)  P(35,2)  P(35,3);P(36,1)  P(36,2)
P(36,3);P(37,1)  P(37,2)  P(37,3);P(38,1)  P(38,2)  P(38,3);P(39,1)  P(39,2)
P(39,3);P(40,1)  P(40,2)  P(40,3);P(41,1)  P(41,2)  P(41,3);P(42,1)  P(42,2)
P(42,3);P(43,1)  P(43,2)  P(43,3);P(44,1)  P(44,2)  P(44,3);P(45,1)  P(45,2)
P(45,3);P(46,1)  P(46,2)  P(46,3);P(47,1)  P(47,2)  P(47,3);P(48,1)  P(48,2)
P(48,3);P(49,  1)  P(49,2)  P(49,3);P(50,1)  P(50,2)  P(50,3);P(51,1)  P(51,2)
P(51,3);P(52,1)  P(52,2)  P(52,3);P(53,  1)  P(53,2)  P(53,3);P(54,1)  P(54,2)
P(54,3);P(55,1)  P(55,2)  P(55,3);P(56,1)  P(56,2)  P(56,3);P(57,1)  P(57,2)
P(57,3);P(58,1)  P(58,2)  P(58,3);P(59,1)  P(59,2)  P(59,3);P(60,1)  P(60,2)
P(60,3);P(61,1)  P(61,2)  P(61,3);P(62,1)  P(62,2)  P(62,3);P(63,1)  P(63,2)
P(63,3);P(64,1)  P(64,2)  P(64,3);P(65,1)  P(65,2)  P(65,3);P(66,1)  P(66,2)
P(66,3);P(67,1)  P(67,2)  P(67,3);P(68,1)  P(68,2)  P(68,3);P(69,1)  P(69,2)
P(69,3);P(70,1)  P(70,2)  P(70,3);P(71,1)  P(71,2)  P(71,3);P(72,1)  P(72,2)
P(72,3);P(73,1)  P(73,2)  P(73,3);P(74,1)  P(74,2)  P(74,3);P(75,1)  P(75,2)
P(75,3);P(76,1)  P(76,2)  P(76,3);P(77,1)  P(77,2)  P(77,3);P(78,1)  P(78,2)
P(78,3);P(79,1)  P(79,2)  P(79,3);P(80,1)  P(80,2)  P(80,3);P(81,1)  P(81,2)
P(81,3);P(82,1)  P(82,2)  P(82,3);P(83,1)  P(83,2)  P(83,3);P(84,1)  P(84,2)
P(84,3);P(85,1)  P(85,2)  P(85,3);P(86,1)  P(86,2)  P(86,3);P(87,1)  P(87,2)
P(87,3);P(88,1)  P(88,2)  P(88,3);P(89,1)  P(89,2)  P(89,3);P(90,1)  P(90,2)
P(90,3);P(91,1)  P(91,2)  P(91,3);P(92,1)  P(92,2)  P(92,3);P(93,1)  P(93,2)
```

```
P(93,3);P(94,1)  P(94,2)  P(94,3);P(95,1)  P(95,2)  P(95,3);P(96,1)  P(96,2)
P(96,3);P(97,1)  P(97,2)  P(97,3);P(98,1)  P(98,2)  P(98,3);P(99,1)  P(99,2)
P(99,3);P(100,1) P(100,2) P(100,3);P( 101,1) P(101,2) P(101,3);P(102, 1)
P(102,2)    P(102,3);P(103,1)    P(103,2)    P(103,3);P(104,1)    P(104,2)
P(104,3);P(105,1) P(105,2) P(105,3);P(106,1) P(106,2) P(106,3);P(107, 1)
P(107,2)    P(107,3);P(108,1)    P(108,2)    P(108,3);P(109,1)    P(109,2)
P(109,3);P(110,1)  P(110,2)  P(110,3);P(111,1)  P(111,2)  P(111,3);P(112,1)
P(112,2)    P(112,3);P(113,1)    P(113,2)    P(113,3);P(114,1)    P(114,2)
P(114,3);P(115,1)  P(115,2)  P(115,3);P(116,1)  P(116,2)  P(116,3);P(117,1)
P(117,2)    P(117,3);P(118,1)    P(118,2)    P(118,3);P(119,1)    P(119,2)
P(119,3);P(120,1)  P(120,2)  P(120,3);P(121,1)  P(121,2)  P(121,3);P(122,1)
P(122,2)    P(122,3);P(123,1)    P(123,2)    P(123,3);P(124,1)    P(124,2)
P(124,3);P(125,1)  P(125,2)  P(125,3);P(126,1)  P(126,2)  P(126,3);P(127,1)
P(127,2)    P(127,3);P(128,1)    P(128,2)    P(128,3);P(129,1)    P(129,2)
P(129,3);P(130,1)  P(130,2)  P(130,3);P(131,1)  P(131,2)  P(131,3);P(132,1)
P(132,2)    P(132,3);P(133,1)    P(133,2)    P(133,3);P(134,1)    P(134,2)
P(134,3);P(135,1)  P(135,2)  P(135,3);P(136,1)  P(136,2)  P(136,3);P(137,1)
P(137,2)    P(137,3);P(138,1)    P(138,2)    P(138,3);P(139,1)    P(139,2)
P(139,3);P(140,1)  P(140,2)  P(140,3);P(141,1)  P(141,2)  P(141,3);P(142,1)
P(142,2)    P(142,3);P(143,1)    P(143,2)    P(143,3);P(144,1)    P(144,2)
P(144,3);P(145,1)  P(145,2)  P(145,3);P(146,1)  P(146,2)  P(146,3);P(147,1)
P(147,2)   P(147,3);P(148,1)   P(148,2)   P(148,3);P(149,   1)   P(149,2)
P(149,3);P(150,1)  P(150,2)  P(150,3);P(151,1)  P(151,2)  P(151,3);P(152,1)
P(152,2)   P(152,3);P(153,   1)   P(153,2)   P(153,3);P(154,1)   P(154,2)
P(154,3);P(155,1)  P(155,2)  P(155,3);P(156,1)  P(156,2)  P(156,3);P(157,1)
P(157,2)    P(157,3);P(158,1)    P(158,2)    P(158,3);P(159,1)    P(159,2)
P(159,3);P(160,1)  P(160,2)  P(160,3);P(161,1)  P(161,2)  P(161,3);P(162,1)
P(162,2)    P(162,3);P(163,1)    P(163,2)    P(63,3);P(164,1)    P(164,2)
P(164,3);P(165,1)  P(165,2)  P(165,3);P(166,1)  P(166,2)  P(166,3);P(167,1)
P(167,2)    P(167,3);P(168,1)    P(168,2)    P(168,3);P(169,1)    P(169,2)
P(169,3);P(170,1)  P(170,2)  P(170,3);P(171,1)  P(171,2)  P(171,3);P(172,1)
P(172,2)    P(172,3);P(173,1)    P(173,2)    P(173,3);P(174,1)    P(174,2)
P(174,3);P(175,1)  P(175,2)  P(175,3);P(176,1)  P(176,2)  P(176,3);P(177,1)
P(177,2)    P(177,3);P(178,1)    P(178,2)    P(178,3);P(179,1)    P(179,2)
P(179,3);P(180,1)  P(180,2)  P(180,3);P(181,1)  P(181,2)  P(181,3);P(182,1)
P(182,2)    P(182,3);P(183,1)    P(183,2)    P(183,3);P(184,1)    P(184,2)
P(184,3);P(185,1)  P(185,2)  P(185,3);P(186,1)  P(186,2)  P(186,3);P(187,1)
P(187,2)    P(187,3);P(188,1)    P(188,2)    P(188,3);P(189,1)    P(189,2)
P(189,3);P(190,1)  P(190,2)  P(190,3);P(191,1)  P(191,2)  P(191,3);P(192,1)
P(192,2)    P(192,3);P(193,1)    P(193,2)    P(193,3);P(194,1)    P(194,2)
P(194,3);P(195,1)  P(195,2)  P(195,3);P(196,1)  P(196,2)  P(196,3);P(197,1)
P(197,2)    P(197,3);P(198,1)    P(198,2)    P(198,3);P(199,1)    P(199,2)
P(199,3);P(200,1) P(200,2) P(200,3);P( 201,1) P(201,2) P(201,3);P(202, 1)
P(202,2)    P(202,3);P(203,1)    P(203,2)    P(203,3);P(204,1)    P(204,2)
P(204,3);P(205,1) P(205,2) P(205,3);P(206,1) P(206,2) P(206,3);P(207, 1)
P(207,2)    P(207,3);P(208,1)    P(208,2)    P(208,3);P(209,1)    P(209,2)
P(209,3);P(210,1)  P(210,2)  P(210,3);P(211,1)  P(211,2)  P(211,3);P(212,1)
P(212,2)    P(212,3);P(213,1)    P(213,2)    P(213,3);P(214,1)    P(214,2)
P(214,3);P(215,1)  P(215,2)  P(215,3);P(216,1)  P(216,2)  P(216,3);P(217,1)
P(217,2)    P(217,3);P(218,1)    P(218,2)    P(218,3);P(219,1)    P(219,2)
P(219,3);P(220,1)  P(220,2)  P(220,3);P(221,1)  P(221,2)  P(221,3);P(222,1)
P(222,2)    P(222,3);P(223,1)    P(223,2)    P(223,3);P(224,1)    P(224,2)
P(224,3);P(225,1)  P(225,2)  P(225,3);P(226,1)  P(226,2)  P(226,3);P(227,1)
P(227,2)    P(227,3);P(228,1)    P(228,2)    P(228,3);P(229,1)    P(229,2)
P(229,3);P(230,1)  P(230,2)  P(230,3);P(231,1)  P(231,2)  P(231,3);P(232,1)
P(232,2)    P(232,3);P(233,1)    P(233,2)    P(233,3);P(234,1)    P(234,2)
P(234,3);P(235,1)  P(235,2)  P(235,3);P(236,1)  P(236,2)  P(236,3);P(237,1)
```

```
P(237,2)    P(237,3);P(238,1)    P(238,2)    P(238,3);P(239,1)    P(239,2)
P(239,3);P(240,1)  P(240,2)  P(240,3);P(241,1)  P(241,2)  P(241,3);P(242,1)
P(242,2)    P(242,3);P(243,1)    P(243,2)    P(243,3);P(244,1)    P(244,2)
P(244,3);P(245,1)  P(245,2)  P(245,3);P(246,1)  P(246,2)  P(246,3);P(247,1)
P(247,2)   P(247,3);P(248,1)   P(248,2)   P(248,3);P(249,   1)   P(249,2)
P(249,3);P(250,1)  P(250,2)  P(250,3);P(251,1)  P(251,2)  P(251,3);P(252,1)
P(252,2)   P(252,3);P(253,   1)   P(253,2)   P(253,3);P(254,1)   P(254,2)
P(254,3);P(255,1)  P(255,2)  P(255,3);P(256,1)  P(256,2)  P(256,3);P(257,1)
P(257,2)    P(257,3);P(258,1)    P(258,2)    P(258,3);P(259,1)    P(259,2)
P(259,3);P(260,1)  P(260,2)  P(260,3);P(261,1)  P(261,2)  P(261,3);P(262,1)
P(262,2)    P(262,3);P(263,1)    P(263,2)    P(263,3);P(264,1)    P(264,2)
P(264,3);P(265,1)  P(265,2)  P(265,3);P(266,1)  P(266,2)  P(266,3);P(267,1)
P(267,2)    P(267,3);P(268,1)    P(268,2)    P(268,3);P(269,1)    P(269,2)
P(269,3);P(270,1)  P(270,2)  P(270,3);P(271,1)  P(271,2)  P(271,3);P(272,1)
P(272,2)    P(272,3);P(273,1)    P(273,2)    P(273,3);P(274,1)    P(274,2)
P(274,3);P(275,1)  P(275,2)  P(275,3);P(276,1)  P(276,2)  P(276,3);P(277,1)
P(277,2)    P(277,3);P(278,1)    P(278,2)    P(278,3);P(279,1)    P(279,2)
P(279,3);P(280,1)  P(280,2)  P(280,3);P(281,1)  P(281,2)  P(281,3);P(282,1)
P(282,2)    P(282,3);P(283,1)    P(283,2)    P(283,3);P(284,1)    P(284,2)
P(284,3);P(285,1)  P(285,2)  P(285,3);P(286,1)  P(286,2)  P(286,3);P(287,1)
P(287,2)    P(287,3);P(288,1)    P(288,2)    P(288,3);P(289,1)    P(289,2)
P(289,3);P(290,1)  P(290,2)  P(290,3);P(291,1)  P(291,2)  P(291,3);P(292,1)
P(292,2)    P(292,3);P(293,1)    P(293,2)    P(293,3);P(294,1)    P(294,2)
P(294,3);P(295,1)  P(295,2)  P(295,3);P(296,1)  P(296,2)  P(296,3);P(297,1)
P(297,2)    P(297,3);P(298,1)    P(298,2)    P(298,3);P(299,1)    P(299,2)
P(299,3);P(300,1)  P(300,2)  P(300,3);P(301,1)  P(301,2)  P(301,3);P(302,  1)
P(302,2)    P(302,3);P(303,1)    P(303,2)    P(303,3);P(304,1)    P(304,2)
P(304,3);P(305,1)  P(305,2)  P(305,3);P(306,1)  P(306,2)  P(306,3);P(307,  1)
P(307,2)    P(307,3);P(308,1)    P(308,2)    P(308,3);P(309,1)    P(309,2)
P(309,3);P(310,1)  P(310,2)  P(310,3);P(311,1)  P(311,2)  P(311,3);P(312,1)
P(312,2)    P(312,3);P(313,1)    P(313,2)    P(313,3);P(314,1)    P(314,2)
P(314,3);P(315,1)  P(315,2)  P(315,3);P(316,1)  P(316,2)  P(316,3);P(317,1)
P(317,2)    P(317,3);P(318,1)    P(318,2)    P(318,3);P(319,1)    P(319,2)
P(319,3);P(320,1)  P(320,2)  P(320,3);P(321,1)  P(321,2)  P(321,3);P(322,1)
P(322,2)    P(322,3);P(323,1)    P(323,2)    P(323,3);P(324,1)    P(324,2)
P(324,3);P(325,1)  P(325,2)  P(325,3);P(326,1)  P(326,2)  P(326,3);P(327,1)
P(327,2)    P(327,3);P(328,1)    P(328,2)    P(328,3);P(329,1)    P(329,2)
P(329,3);P(330,1)  P(330,2)  P(330,3);P(331,1)  P(331,2)  P(331,3);P(332,1)
P(332,2)    P(332,3);P(333,1)    P(333,2)    P(333,3);P(334,1)    P(334,2)
P(334,3);P(335,1)  P(335,2)  P(335,3);P(336,1)  P(336,2)  P(336,3);P(337,1)
P(337,2)    P(337,3);P(338,1)    P(338,2)    P(338,3);P(339,1)    P(339,2)
P(339,3);P(340,1)  P(340,2)  P(340,3);P(341,1)  P(341,2)  P(341,3);P(342,1)
P(342,2)    P(342,3);P(343,1)    P(343,2)    P(343,3);P(344,1)    P(344,2)
P(344,3);P(345,1)  P(345,2)  P(345,3);P(346,1)  P(346,2)  P(346,3);P(347,1)
P(347,2)   P(347,3);P(348,1)   P(348,2)   P(348,3);P(349,   1)   P(349,2)
P(349,3);P(350,1)  P(350,2)  P(350,3);P(351,1)  P(351,2)  P(351,3);P(352,1)
P(352,2)   P(352,3);P(353,   1)   P(353,2)   P(353,3);P(354,1)   P(354,2)
P(354,3);P(355,1)  P(355,2)  P(355,3);P(356,1)  P(356,2)  P(356,3);P(357,1)
P(357,2)    P(357,3);P(358,1)    P(358,2)    P(358,3);P(359,1)    P(359,2)
P(359,3);P(360,1)  P(360,2)  P(360,3);P(361,1)  P(361,2)  P(361,3);P(362,1)
P(362,2)    P(362,3);P(363,1)    P(363,2)    P(363,3);P(364,1)    P(364,2)
P(364,3);P(365,1)  P(365,2)  P(365,3);P(366,1)  P(366,2)  P(366,3);P(367,1)
P(367,2)    P(367,3);P(368,1)    P(368,2)    P(368,3);P(369,1)    P(369,2)
P(369,3);P(370,1)  P(370,2)  P(370,3);P(371,1)  P(371,2)  P(371,3);P(372,1)
P(372,2)    P(372,3);P(373,1)    P(373,2)    P(373,3);P(374,1)    P(374,2)
P(374,3);P(375,1)  P(375,2)  P(375,3);P(376,1)  P(376,2)  P(376,3);P(377,1)
P(377,2)    P(377,3);P(378,1)    P(378,2)    P(378,3);P(379,1)    P(379,2)
```

```
P(379,3);P(380,1)  P(380,2)  P(380,3);P(381,1)  P(381,2)  P(381,3);P(382,1)
P(382,2)    P(382,3);P(383,1)    P(383,2)    P(383,3);P(384,1)    P(384,2)
P(384,3);P(385,1)  P(385,2)  P(385,3);P(386,1)  P(386,2)  P(386,3);P(387,1)
P(387,2)    P(387,3);P(388,1)    P(388,2)    P(388,3);P(389,1)    P(389,2)
P(389,3);P(390,1)  P(390,2)  P(390,3);P(391,1)  P(391,2)  P(391,3);P(392,1)
P(392,2)    P(392,3);P(393,1)    P(393,2)    P(393,3);P(394,1)    P(394,2)
P(394,3);P(395,1)  P(395,2)  P(395,3);P(396,1)  P(396,2)  P(396,3);P(397,1)
P(397,2)    P(397,3);P(398,1)    P(398,2)    P(398,3);P(399,1)    P(399,2)
P(399,3);P(400,1) P(400,2) P(400,3);P( 401,1) P(401,2) P(401,3);P(402, 1)
P(402,2)    P(402,3);P(403,1)    P(403,2)    P(403,3);P(404,1)    P(404,2)
P(404,3);P(405,1) P(405,2) P(405,3);P(406,1) P(406,2) P(406,3);P(407, 1)
P(407,2)    P(407,3);P(408,1)    P(408,2)    P(408,3);P(409,1)    P(409,2)
P(409,3);P(410,1)  P(410,2)  P(410,3);P(411,1)  P(411,2)  P(411,3);P(412,1)
P(412,2)    P(412,3);P(413,1)    P(413,2)    P(413,3);P(414,1)    P(414,2)
P(414,3);P(415,1)  P(415,2)  P(415,3);P(416,1)  P(416,2)  P(416,3);P(417,1)
P(417,2)    P(417,3);P(418,1)    P(418,2)    P(418,3);P(419,1)    P(419,2)
P(419,3);P(420,1)  P(420,2)  P(420,3);P(421,1)  P(421,2)  P(421,3);P(422,1)
P(422,2)    P(422,3);P(423,1)    P(423,2)    P(423,3);P(424,1)    P(424,2)
P(424,3);P(425,1)  P(425,2)  P(425,3);P(426,1)  P(426,2)  P(426,3);P(427,1)
P(427,2)    P(427,3);P(428,1)    P(428,2)    P(428,3);P(429,1)    P(429,2)
P(429,3);P(430,1)  P(430,2)  P(430,3);P(431,1)  P(431,2)  P(431,3);P(432,1)
P(432,2)    P(432,3);P(433,1)    P(433,2)    P(433,3);P(434,1)    P(434,2)
P(434,3);P(435,1)  P(435,2)  P(435,3);P(436,1)  P(436,2)  P(436,3);P(437,1)
P(437,2)    P(437,3);P(438,1)    P(438,2)    P(438,3);P(439,1)    P(439,2)
P(439,3);P(440,1)  P(440,2)  P(440,3);P(441,1)  P(441,2)  P(441,3);P(442,1)
P(442,2)    P(442,3);P(443,1)    P(443,2)    P(443,3);P(444,1)    P(444,2)
P(444,3);P(445,1)  P(445,2)  P(445,3);P(446,1)  P(446,2)  P(446,3);P(447,1)
P(447,2)   P(447,3);P(448,1)   P(448,2)   P(448,3);P(449,   1)   P(449,2)
P(449,3);P(450,1)  P(450,2)  P(450,3);P(451,1)  P(451,2)  P(451,3);P(452,1)
P(452,2)   P(452,3);P(453,   1)   P(453,2)   P(453,3);P(454,1)   P(454,2)
P(454,3);P(455,1)  P(455,2)  P(455,3);P(456,1)  P(456,2)  P(456,3);P(457,1)
P(457,2)    P(457,3);P(458,1)    P(458,2)    P(458,3);P(459,1)    P(459,2)
P(459,3);P(460,1)  P(460,2)  P(460,3);P(461,1)  P(461,2)  P(461,3);P(462,1)
P(462,2)    P(462,3);P(463,1)    P(463,2)    P(63,3);P(464,1)    P(464,2)
P(464,3);P(465,1)  P(465,2)  P(465,3);P(466,1)  P(466,2)  P(466,3);P(467,1)
P(467,2)    P(467,3);P(468,1)    P(468,2)    P(468,3);P(469,1)    P(469,2)
P(469,3);P(470,1)  P(470,2)  P(470,3);P(471,1)  P(471,2)  P(471,3);P(472,1)
P(472,2)    P(472,3);P(473,1)    P(473,2)    P(473,3);P(474,1)    P(474,2)
P(474,3);P(475,1)  P(475,2)  P(475,3);P(476,1)  P(476,2)  P(476,3);P(477,1)
P(477,2)    P(477,3);P(478,1)    P(478,2)    P(478,3);P(479,1)    P(479,2)
P(479,3);P(480,1)  P(480,2)  P(480,3);P(481,1)  P(481,2)  P(481,3);P(482,1)
P(482,2)    P(482,3);P(483,1)    P(483,2)    P(483,3);P(484,1)    P(484,2)
P(484,3);P(485,1)  P(485,2)  P(485,3);P(486,1)  P(486,2)  P(486,3);P(487,1)
P(487,2)    P(487,3);P(488,1)    P(488,2)    P(488,3);P(489,1)    P(489,2)
P(489,3);P(490,1)  P(490,2)  P(490,3);P(491,1)  P(491,2)  P(491,3);P(492,1)
P(492,2)    P(492,3);P(493,1)    P(493,2)    P(493,3);P(494,1)    P(494,2)
P(494,3);P(495,1)  P(495,2)  P(495,3);P(496,1)  P(496,2)  P(496,3);P(497,1)
P(497,2)    P(497,3);P(498,1)    P(498,2)    P(498,3);P(499,1)    P(499,2)
P(499,3);P(500,1) P(500,2) P(500,3);P( 501,1) P(501,2) P(501,3);P(502, 1)
P(502,2)    P(502,3);P(503,1)    P(503,2)    P(503,3);P(504,1)    P(504,2)
P(504,3);P(505,1) P(505,2) P(505,3);P(506,1) P(506,2) P(506,3);P(507, 1)
P(507,2)    P(507,3);P(508,1)    P(508,2)    P(508,3);P(509,1)    P(509,2)
P(509,3);P(510,1)  P(510,2)  P(510,3);P(511,1)  P(511,2)  P(511,3);P(512,1)
P(512,2)    P(512,3);P(513,1)    P(513,2)    P(513,3);P(514,1)    P(514,2)
P(514,3);P(515,1)  P(515,2)  P(515,3);P(516,1)  P(516,2)  P(516,3);P(517,1)
P(517,2)    P(517,3);P(518,1)    P(518,2)    P(518,3);P(519,1)    P(519,2)
P(519,3);P(520,1)  P(520,2)  P(520,3);P(521,1)  P(521,2)  P(521,3);P(522,1)
```

```
P(522,2)    P(522,3);P(523,1)    P(523,2)    P(523,3);P(524,1)    P(524,2)
P(524,3);P(525,1)  P(525,2)  P(525,3);P(526,1)  P(526,2)  P(526,3);P(527,1)
P(527,2)    P(527,3);P(528,1)    P(528,2)    P(528,3);P(529,1)    P(529,2)
P(529,3);P(530,1)  P(530,2)  P(530,3);P(531,1)  P(531,2)  P(531,3);P(532,1)
P(532,2)    P(532,3);P(533,1)    P(533,2)    P(533,3);P(534,1)    P(534,2)
P(534,3);P(535,1)  P(535,2)  P(535,3);P(536,1)  P(536,2)  P(536,3);P(537,1)
P(537,2)    P(537,3);P(538,1)    P(538,2)    P(538,3);P(539,1)    P(539,2)
P(539,3);P(540,1)  P(540,2)  P(540,3);P(541,1)  P(541,2)  P(541,3);P(542,1)
P(542,2)    P(542,3);P(543,1)    P(543,2)    P(543,3);P(544,1)    P(544,2)
P(544,3);P(545,1)  P(545,2)  P(545,3);P(546,1)  P(546,2)  P(546,3);P(547,1)
P(547,2)    P(547,3);P(548,1)    P(548,2)    P(548,3);P(549,1)    P(549,2)
P(549,3);P(550,1)  P(550,2)  P(550,3);P(551,1)  P(551,2)  P(551,3);P(552,1)
P(552,2)    P(552,3);P(553,1)    P(553,2)    P(553,3);P(554,1)    P(554,2)
P(554,3);P(555,1)  P(555,2)  P(555,3);P(556,1)  P(556,2)  P(556,3);P(557,1)
P(557,2)    P(557,3);P(558,1)    P(558,2)    P(558,3);P(559,1)    P(559,2)
P(559,3);P(560,1)  P(560,2)  P(560,3);P(561,1)  P(561,2)  P(561,3);P(562,1)
P(562,2)    P(562,3);P(563,1)    P(563,2)    P(563,3);P(564,1)    P(564,2)
P(564,3);P(565,1)  P(565,2)  P(565,3);P(566,1)  P(566,2)  P(566,3);P(567,1)
P(567,2)    P(567,3);P(568,1)    P(568,2)    P(568,3);P(569,1)    P(569,2)
P(569,3);P(570,1)  P(570,2)  P(570,3);P(571,1)  P(571,2)  P(571,3);P(572,1)
P(572,2)    P(572,3);P(573,1)    P(573,2)    P(573,3);P(574,1)    P(574,2)
P(574,3);P(575,1)  P(575,2)  P(575,3);P(576,1)  P(576,2)  P(576,3);P(577,1)
P(577,2)    P(577,3);P(578,1)    P(578,2)    P(578,3);P(579,1)    P(579,2)
P(579,3);P(580,1)  P(580,2)  P(580,3);P(581,1)  P(581,2)  P(581,3);P(582,1)
P(582,2)    P(582,3);P(583,1)    P(583,2)    P(583,3);P(584,1)    P(584,2)
P(584,3);P(585,1)  P(585,2)  P(585,3);P(586,1)  P(586,2)  P(586,3);P(587,1)
P(587,2)    P(587,3);P(588,1)    P(588,2)    P(588,3);P(589,1)    P(589,2)
P(589,3);P(590,1)  P(590,2)  P(590,3);P(591,1)  P(591,2)  P(591,3);P(592,1)
P(592,2)    P(592,3);P(593,1)    P(593,2)    P(593,3);P(594,1)    P(594,2)
P(594,3);P(595,1)  P(595,2)  P(595,3);P(596,1)  P(596,2)  P(596,3);P(597,1)
P(597,2)    P(597,3);P(598,1)    P(598,2)    P(598,3);P(599,1)    P(599,2)
P(599,3);P(600,1)  P(600,2)  P(600,3);P(601,1)  P(601,2)  P(601,3);P(602,1)
P(602,2)    P(602,3);P(603,1)    P(603,2)    P(603,3);P(604,1)    P(604,2)
P(604,3);P(605,1)  P(605,2)  P(605,3);P(606,1)  P(606,2)  P(606,3);P(607,1)
P(607,2)    P(607,3);P(608,1)    P(608,2)    P(608,3);P(609,1)    P(609,2)
P(609,3);P(610,1)  P(610,2)  P(610,3);P(611,1)  P(611,2)  P(611,3);P(612,1)
P(612,2)    P(612,3);P(613,1)    P(613,2)    P(613,3);P(614,1)    P(614,2)
P(614,3);P(615,1)  P(615,2)  P(615,3);P(616,1)  P(616,2)  P(616,3);P(617,1)
P(617,2)    P(617,3);P(618,1)    P(618,2)    P(618,3);P(619,1)    P(619,2)
P(619,3);P(620,1)  P(620,2)  P(620,3);P(621,1)  P(621,2)  P(621,3);P(622,1)
P(622,2)    P(622,3);P(623,1)    P(623,2)    P(623,3);P(624,1)    P(624,2)
P(624,3);P(625,1)  P(625,2)  P(625,3);P(626,1)  P(626,2)  P(626,3);P(627,1)
P(627,2)    P(627,3);P(628,1)    P(628,2)    P(628,3);P(629,1)    P(629,2)
P(629,3);P(630,1)  P(630,2)  P(630,3);P(631,1)  P(631,2)  P(631,3);P(632,1)
P(632,2)    P(632,3);P(633,1)    P(633,2)    P(633,3);P(634,1)    P(634,2)
P(634,3);P(635,1)  P(635,2)  P(635,3);P(636,1)  P(636,2)  P(636,3);P(637,1)
P(637,2)    P(637,3);P(638,1)    P(638,2)    P(638,3);P(639,1)    P(639,2)
P(639,3);P(640,1)  P(640,2)  P(640,3);P(641,1)  P(641,2)  P(641,3);P(642,1)
P(642,2)    P(642,3);P(643,1)    P(643,2)    P(643,3);P(644,1)    P(644,2)
P(644,3);P(645,1)  P(645,2)  P(645,3);P(646,1)  P(646,2)  P(646,3);P(647,1)
P(647,2)    P(647,3);P(648,1)    P(648,2)    P(648,3);P(649,1)    P(649,2)
P(649,3);P(650,1)  P(650,2)  P(650,3);P(651,1)  P(651,2)  P(651,3);P(652,1)
P(652,2)    P(652,3);P(653,1)    P(653,2)    P(653,3);P(654,1)    P(654,2)
P(654,3);P(655,1)  P(655,2)  P(655,3);P(656,1)  P(656,2)  P(656,3);P(657,1)
P(657,2)    P(657,3);P(658,1)    P(658,2)    P(658,3);P(659,1)    P(659,2)
P(659,3);P(660,1)  P(660,2)  P(660,3);P(661,1)  P(661,2)  P(661,3);P(662,1)
P(662,2)    P(662,3);P(663,1)    P(663,2)    P(663,3);P(664,1)    P(664,2)
```

```
P(664,3);P(665,1)  P(665,2)  P(665,3);P(666,1)  P(666,2)  P(666,3);P(667,1)
P(667,2)    P(667,3);P(668,1)    P(668,2)    P(668,3);P(669,1)    P(669,2)
P(669,3);P(670,1)  P(670,2)  P(670,3);P(671,1)  P(671,2)  P(671,3);P(672,1)
P(672,2)    P(672,3);P(673,1)    P(673,2)    P(673,3);P(674,1)    P(674,2)
P(674,3);P(675,1)  P(675,2)  P(675,3);P(676,1)  P(676,2)  P(676,3);P(677,1)
P(677,2)    P(677,3);P(678,1)    P(678,2)    P(678,3);P(679,1)    P(679,2)
P(679,3);P(680,1)  P(680,2)  P(680,3);P(681,1)  P(681,2)  P(681,3);P(682,1)
P(682,2)    P(682,3);P(683,1)    P(683,2)    P(683,3);P(684,1)    P(684,2)
P(684,3);P(685,1)  P(685,2)  P(685,3);P(686,1)  P(686,2)  P(686,3);P(687,1)
P(687,2)    P(687,3);P(688,1)    P(688,2)    P(688,3);P(689,1)    P(689,2)
P(689,3);P(690,1)  P(690,2)  P(690,3);P(691,1)  P(691,2)  P(691,3);P(692,1)
P(692,2)    P(692,3);P(693,1)    P(693,2)    P(693,3);P(694,1)    P(694,2)
P(694,3);P(695,1)  P(695,2)  P(695,3);P(696,1)  P(696,2)  P(696,3);P(697,1)
P(697,2)    P(697,3);P(698,1)    P(698,2)    P(698,3);P(699,1)    P(699,2)
P(699,3);P(700,1)  P(700,2)  P(700,3);P(701,1)  P(701,2)  P(701,3);P(702,1)
P(702,2)    P(702,3);P(703,1)    P(703,2)    P(703,3);P(704,1)    P(704,2)
P(704,3);P(705,1)  P(705,2)  P(705,3);P(706,1)  P(706,2)  P(706,3);P(707,1)
P(707,2)    P(707,3);P(708,1)    P(708,2)    P(708,3);P(709,1)    P(709,2)
P(709,3);P(710,1)  P(710,2)  P(710,3);P(711,1)  P(711,2)  P(711,3);P(712,1)
P(712,2)    P(712,3);P(713,1)    P(713,2)    P(713,3);P(714,1)    P(714,2)
P(714,3);P(715,1)  P(715,2)  P(715,3);P(716,1)  P(716,2)  P(716,3);P(717,1)
P(717,2)    P(717,3);P(718,1)    P(718,2)    P(718,3);P(719,1)    P(719,2)
P(719,3);P(720,1)  P(720,2)  P(720,3);P(721,1)  P(721,2)  P(721,3);P(722,1)
P(722,2)    P(722,3);P(723,1)    P(723,2)    P(723,3);P(724,1)    P(724,2)
P(724,3);P(725,1)  P(725,2)  P(725,3);P(726,1)  P(726,2)  P(726,3);P(727,1)
P(727,2)    P(727,3);P(728,1)    P(728,2)    P(728,3);P(729,1)    P(729,2)
P(729,3);P(730,1)  P(730,2)  P(730,3);P(731,1)  P(731,2)  P(731,3);P(732,1)
P(732,2)    P(732,3);P(733,1)    P(733,2)    P(733,3);P(734,1)    P(734,2)
P(734,3);P(735,1)  P(735,2)  P(735,3);P(736,1)  P(736,2)  P(736,3);P(737,1)
P(737,2)    P(737,3);P(738,1)    P(738,2)    P(738,3);P(739,1)    P(739,2)
P(739,3);P(740,1)  P(740,2)  P(740,3);P(741,1)  P(741,2)  P(741,3);P(742,1)
P(742,2)    P(742,3);P(743,1)    P(743,2)    P(743,3);P(744,1)    P(744,2)
P(744,3);P(745,1)  P(745,2)  P(745,3);P(746,1)  P(746,2)  P(746,3);P(747,1)
P(747,2)   P(747,3);P(748,1)   P(748,2)   P(748,3);P(749,   1)   P(749,2)
P(749,3);P(750,1)  P(750,2)  P(750,3);P(751,1)  P(751,2)  P(751,3);P(752,1)
P(752,2)    P(752,3);P(753,1)    P(753,2)    P(753,3);P(754,1)    P(754,2)
P(754,3);P(755,1)  P(755,2)  P(755,3);P(756,1)  P(756,2)  P(756,3);P(757,1)
P(757,2)    P(757,3);P(758,1)    P(758,2)    P(758,3);P(759,1)    P(759,2)
P(759,3);P(760,1)  P(760,2)  P(760,3);P(761,1)  P(761,2)  P(761,3);P(762,1)
P(762,2)    P(762,3);P(763,1)    P(763,2)    P(763,3);P(764,1)    P(764,2)
P(764,3);P(765,1)  P(765,2)  P(765,3);P(766,1)  P(766,2)  P(766,3);P(767,1)
P(767,2)    P(767,3);P(768,1)    P(768,2)    P(768,3);P(769,1)    P(769,2)
P(769,3);P(770,1)  P(770,2)  P(770,3);P(771,1)  P(771,2)  P(771,3);P(772,1)
P(772,2)    P(772,3);P(773,1)    P(773,2)    P(773,3);P(774,1)    P(774,2)
P(774,3);P(775,1)  P(775,2)  P(775,3);P(776,1)  P(776,2)  P(776,3);P(777,1)
P(777,2)    P(777,3);P(778,1)    P(778,2)    P(778,3);P(779,1)    P(779,2)
P(779,3);P(780,1)  P(780,2)  P(780,3);P(781,1)  P(781,2)  P(781,3);P(782,1)
P(782,2)    P(782,3);P(783,1)    P(783,2)    P(783,3);P(784,1)    P(784,2)
P(784,3);P(785,1)  P(785,2)  P(785,3);P(786,1)  P(786,2)  P(786,3);P(787,1)
P(787,2)    P(787,3);P(788,1)    P(788,2)    P(788,3);P(789,1)    P(789,2)
P(789,3);P(790,1)  P(790,2)  P(790,3);P(791,1)  P(791,2)  P(791,3);P(792,1)
P(792,2)    P(792,3);P(793,1)    P(793,2)    P(793,3);P(794,1)    P(794,2)
P(794,3);P(795,1)  P(795,2)  P(795,3);P(796,1)  P(796,2)  P(796,3);P(797,1)
P(797,2)    P(797,3);P(798,1)    P(798,2)    P(798,3);P(799,1)    P(799,2)
P(799,3);P(800,1)  P(800,2)  P(800,3);P( 801,1)  P(801,2)  P(801,3);P(802,1)
P(802,2)    P(802,3);P(803,1)    P(803,2)    P(803,3);P(804,1)    P(804,2)
P(804,3);P(805,1)  P(805,2)  P(805,3);P(806,1)  P(806,2)  P(806,3);P(807,1)
```

```
P(807,2)    P(807,3);P(808,1)    P(808,2)    P(808,3);P(809,1)    P(809,2)
P(809,3);P(810,1)  P(810,2)  P(810,3);P(811,1)  P(811,2)  P(811,3);P(812,1)
P(812,2)    P(812,3);P(813,1)    P(813,2)    P(813,3);P(814,1)    P(814,2)
P(814,3);P(815,1)  P(815,2)  P(815,3);P(816,1)  P(816,2)  P(816,3);P(817,1)
P(817,2)    P(817,3);P(818,1)    P(818,2)    P(818,3);P(819,1)    P(819,2)
P(819,3);P(820,1)  P(820,2)  P(820,3);P(821,1)  P(821,2)  P(821,3);P(822,1)
P(822,2)    P(822,3);P(823,1)    P(823,2)    P(823,3);P(824,1)    P(824,2)
P(824,3);P(825,1)  P(825,2)  P(825,3);P(826,1)  P(826,2)  P(826,3);P(827,1)
P(827,2)    P(827,3);P(828,1)    P(828,2)    P(828,3);P(829,1)    P(829,2)
P(829,3);P(830,1)  P(830,2)  P(830,3);P(831,1)  P(831,2)  P(831,3);P(832,1)
P(832,2)    P(832,3);P(833,1)    P(833,2)    P(833,3);P(834,1)    P(834,2)
P(834,3);P(835,1)  P(835,2)  P(835,3);P(836,1)  P(836,2)  P(836,3);P(837,1)
P(837,2)    P(837,3);P(838,1)    P(838,2)    P(838,3);P(839,1)    P(839,2)
P(839,3);P(840,1)  P(840,2)  P(840,3);P(841,1)  P(841,2)  P(841,3);P(842,1)
P(842,2)    P(842,3);P(843,1)    P(843,2)    P(843,3);P(844,1)    P(844,2)
P(844,3);P(845,1)  P(845,2)  P(845,3);P(846,1)  P(846,2)  P(846,3);P(847,1)
P(847,2)    P(847,3);P(848,1)    P(848,2)    P(848,3);P(849,1)    P(849,2)
P(849,3);P(850,1)  P(850,2)  P(850,3);P(851,1)  P(851,2)  P(851,3);P(852,1)
P(852,2)    P(852,3);P(853,1)    P(853,2)    P(853,3);P(854,1)    P(854,2)
P(854,3);P(855,1)  P(855,2)  P(855,3);P(856,1)  P(856,2)  P(856,3);P(857,1)
P(857,2)    P(857,3);P(858,1)    P(858,2)    P(858,3);P(859,1)    P(859,2)
P(859,3);P(860,1)  P(860,2)  P(860,3);P(861,1)  P(861,2)  P(861,3);P(862,1)
P(862,2)    P(862,3);P(863,1)    P(863,2)    P(863,3);P(864,1)    P(864,2)
P(864,3);P(865,1)  P(865,2)  P(865,3);P(866,1)  P(866,2)  P(866,3);P(867,1)
P(867,2)    P(867,3);P(868,1)    P(868,2)    P(868,3);P(869,1)    P(869,2)
P(869,3);P(870,1)  P(870,2)  P(870,3);P(871,1)  P(871,2)  P(871,3);P(872,1)
P(872,2)    P(872,3);P(873,1)    P(873,2)    P(873,3);P(874,1)    P(874,2)
P(874,3);P(875,1)  P(875,2)  P(875,3);P(876,1)  P(876,2)  P(876,3);P(877,1)
P(877,2)    P(877,3);P(878,1)    P(878,2)    P(878,3);P(879,1)    P(879,2)
P(879,3);P(880,1)  P(880,2)  P(880,3);P(881,1)  P(881,2)  P(881,3);P(882,1)
P(882,2)    P(882,3);P(883,1)    P(883,2)    P(883,3);P(884,1)    P(884,2)
P(884,3);P(885,1)  P(885,2)  P(885,3);P(886,1)  P(886,2)  P(886,3);P(887,1)
P(887,2)    P(887,3);P(888,1)    P(888,2)    P(888,3);P(889,1)    P(889,2)
P(889,3);P(890,1)  P(890,2)  P(890,3);P(891,1)  P(891,2)  P(891,3);P(892,1)
P(892,2)    P(892,3);P(893,1)    P(893,2)    P(893,3);P(894,1)    P(894,2)
P(894,3);P(895,1)  P(895,2)  P(895,3);P(896,1)  P(896,2)  P(896,3);P(897,1)
P(897,2)    P(897,3);P(898,1)    P(898,2)    P(898,3);P(899,1)    P(899,2)
P(899,3);P(900,1)  P(900,2)  P(900,3);P(901,1)  P(901,2)  P(901,3);P(902,1)
P(902,2)    P(902,3);P(903,1)    P(903,2)    P(903,3);P(904,1)    P(904,2)
P(904,3);P(905,1)  P(905,2)  P(905,3);P(906,1)  P(906,2)  P(906,3);P(907,1)
P(907,2)    P(907,3);P(908,1)    P(908,2)    P(908,3);P(909,1)    P(909,2)
P(909,3);P(910,1)  P(910,2)  P(910,3);P(911,1)  P(911,2)  P(911,3);P(912,1)
P(912,2)    P(912,3);P(913,1)    P(913,2)    P(913,3);P(914,1)    P(914,2)
P(914,3);P(915,1)  P(915,2)  P(915,3);P(916,1)  P(916,2)  P(916,3);P(917,1)
P(917,2)    P(917,3);P(918,1)    P(918,2)    P(918,3);P(919,1)    P(919,2)
P(919,3);P(920,1)  P(920,2)  P(920,3);P(921,1)  P(921,2)  P(921,3);P(922,1)
P(922,2)    P(922,3);P(923,1)    P(923,2)    P(923,3);P(924,1)    P(924,2)
P(924,3);P(925,1)  P(925,2)  P(925,3);P(926,1)  P(926,2)  P(926,3);P(927,1)
P(927,2)    P(927,3);P(928,1)    P(928,2)    P(928,3);P(929,1)    P(929,2)
P(929,3);P(930,1)  P(930,2)  P(930,3);P(931,1)  P(931,2)  P(931,3);P(932,1)
P(932,2)    P(932,3);P(933,1)    P(933,2)    P(933,3);P(934,1)    P(934,2)
P(934,3);P(935,1)  P(935,2)  P(935,3);P(936,1)  P(936,2)  P(936,3);P(937,1)
P(937,2)    P(937,3);P(938,1)    P(938,2)    P(938,3);P(939,1)    P(939,2)
P(939,3);P(940,1)  P(940,2)  P(940,3);P(941,1)  P(941,2)  P(941,3);P(942,1)
P(942,2)    P(942,3);P(943,1)    P(943,2)    P(943,3);P(944,1)    P(944,2)
P(944,3);P(945,1)  P(945,2)  P(945,3);P(946,1)  P(946,2)  P(946,3);P(947,1)
P(947,2)    P(947,3);P(948,1)    P(948,2)    P(948,3);P(949,1)    P(949,2)
```

```
P(949,3);P(950,1)  P(950,2)  P(950,3);P(951,1)  P(951,2)  P(951,3);P(952,1)
P(952,2)    P(952,3);P(953,1)    P(953,2)    P(953,3);P(954,1)    P(954,2)
P(954,3);P(955,1)  P(955,2)  P(955,3);P(956,1)  P(956,2)  P(956,3);P(957,1)
P(957,2)    P(957,3);P(958,1)    P(958,2)    P(958,3);P(959,1)    P(959,2)
P(959,3);P(960,1)  P(960,2)  P(960,3);P(961,1)  P(961,2)  P(961,3);P(962,1)
P(962,2)    P(962,3);P(963,1)    P(963,2)    P(963,3);P(964,1)    P(964,2)
P(964,3);P(965,1)  P(965,2)  P(965,3);P(966,1)  P(966,2)  P(966,3);P(967,1)
P(967,2)    P(967,3);P(968,1)    P(968,2)    P(968,3);P(969,1)    P(969,2)
P(969,3);P(970,1)  P(970,2)  P(970,3);P(971,1)  P(971,2)  P(971,3);P(972,1)
P(972,2)    P(972,3);P(973,1)    P(973,2)    P(973,3);P(974,1)    P(974,2)
P(974,3);P(975,1)  P(975,2)  P(975,3);P(976,1)  P(976,2)  P(976,3);P(977,1)
P(977,2)    P(977,3);P(978,1)    P(978,2)    P(978,3);P(979,1)    P(979,2)
P(979,3);P(980,1)  P(980,2)  P(980,3);P(981,1)  P(981,2)  P(981,3);P(982,1)
P(982,2)    P(982,3);P(983,1)    P(983,2)    P(983,3);P(984,1)    P(984,2)
P(984,3);P(985,1)  P(985,2)  P(985,3);P(986,1)  P(986,2)  P(986,3);P(987,1)
P(987,2)    P(987,3);P(988,1)    P(988,2)    P(988,3);P(989,1)    P(989,2)
P(989,3);P(990,1)  P(990,2)  P(990,3);P(991,1)  P(991,2)  P(991,3);P(992,1)
P(992,2)    P(992,3);P(993,1)    P(993,2)    P(993,3);P(994,1)    P(994,2)
P(994,3);P(995,1)  P(995,2)  P(995,3);P(996,1)  P(996,2)  P(996,3);P(997,1)
P(997,2)    P(997,3);P(998,1)    P(998,2)    P(998,3);P(999,1)    P(999,2)
P(999,3);P(1000,1)      P(1000,2)      P(1000,3);P(1001,1)      P(1001,2)
P(1001,3);P(1002,1)      P(1002,2)      P(1002,3);P(1003,1)      P(1003,2)
P(1003,3);P(1004,1)      P(1004,2)      P(1004,3);P(1005,1)      P(1005,2)
P(1005,3);P(1006,1)      P(1006,2)      P(1006,3);P(1007,1)      P(1007,2)
P(1007,3);P(1008,1)      P(1008,2)      P(1008,3);P(1009,1)      P(1009,2)
P(1009,3);P(1010,1)      P(1010,2)      P(1010,3);P(1011,1)      P(1011,2)
P(1011,3);P(1012,1)      P(1012,2)      P(1012,3);P(1013,1)      P(1013,2)
P(1013,3);P(1014,1)      P(1014,2)      P(1014,3);P(1015,1)      P(1015,2)
P(1015,3);P(1016,1)      P(1016,2)      P(1016,3);P(1017,1)      P(1017,2)
P(1017,3);P(1018,1)      P(1018,2)      P(1018,3);P(1019,1)      P(1019,2)
P(1019,3);P(1020,1)      P(1020,2)      P(1020,3);P(1021,1)      P(1021,2)
P(1021,3);P(1022,1)      P(1022,2)      P(1022,3);P(1023,1)      P(1023,2)
P(1023,3);P(1024,1)      P(1024,2)      P(1024,3);P(1025,1)      P(1025,2)
P(1025,3);P(1026,1)      P(1026,2)      P(1026,3);P(1027,1)      P(1027,2)
P(1027,3);P(1028,1)      P(1028,2)      P(1028,3);P(1029,1)      P(1029,2)
P(1029,3);P(1030,1)      P(1030,2)      P(1030,3);P(1031,1)      P(1031,2)
P(1031,3);P(1032,1)      P(1032,2)      P(1032,3);P(1033,1)      P(1033,2)
P(1033,3);P(1034,1)      P(1034,2)      P(1034,3);P(1035,1)      P(1035,2)
P(1035,3);P(1036,1)      P(1036,2)      P(1036,3);P(1037,1)      P(1037,2)
P(1037,3);P(1038,1)      P(1038,2)      P(1038,3);P(1039,1)      P(1039,2)
P(1039,3);P(1040,1)      P(1040,2)      P(1040,3);P(1041,1)      P(1041,2)
P(1041,3);P(1042,1)      P(1042,2)      P(1042,3);P(1043,1)      P(1043,2)
P(1043,3);P(1044,1)      P(1044,2)      P(1044,3);P(1045,1)      P(1045,2)
P(1045,3);P(1046,1)      P(1046,2)      P(1046,3);P(1047,1)      P(1047,2)
P(1047,3);P(1048,1)      P(1048,2)      P(1048,3);P(1049,1)      P(1049,2)
P(1049,3);P(1050,1)      P(1050,2)      P(1050,3);P(1051,1)      P(1051,2)
P(1051,3);P(1052,1)      P(1052,2)      P(1052,3);P(1053,1)      P(1053,2)
P(1053,3);P(1054,1)      P(1054,2)      P(1054,3);P(1055,1)      P(1055,2)
P(1055,3);P(1056,1)      P(1056,2)      P(1056,3);P(1057,1)      P(1057,2)
P(1057,3);P(1058,1)      P(1058,2)      P(1058,3);P(1059,1)      P(1059,2)
P(1059,3);P(1060,1)      P(1060,2)      P(1060,3);P(1061,1)      P(1061,2)
P(1061,3);P(1062,1)      P(1062,2)      P(1062,3);P(1063,1)      P(1063,2)
P(1063,3);P(1064,1)      P(1064,2)      P(1064,3);P(1065,1)      P(1065,2)
P(1065,3);P(1066,1)      P(1066,2)      P(1066,3);P(1067,1)      P(1067,2)
P(1067,3);P(1068,1)      P(1068,2)      P(1068,3);P(1069,1)      P(1069,2)
P(1069,3);P(1070,1)      P(1070,2)      P(1070,3);P(1071,1)      P(1071,2)
P(1071,3);P(1072,1)      P(1072,2)      P(1072,3);P(1073,1)      P(1073,2)
```

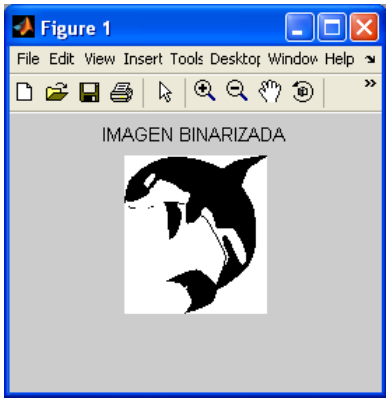
```
P(1073,3);P(1074,1)     P(1074,2)     P(1074,3);P(1075,1)     P(1075,2)
P(1075,3);P(1076,1)     P(1076,2)     P(1076,3);P(1077,1)     P(1077,2)
P(1077,3);P(1078,1)     P(1078,2)     P(1078,3);P(1079,1)     P(1079,2)
P(1079,3);P(1080,1)     P(1080,2)     P(1080,3);P(1081,1)     P(1081,2)
P(1081,3);P(1082,1)     P(1082,2)     P(1082,3);P(1083,1)     P(1083,2)
P(1083,3);P(1084,1)     P(1084,2)     P(1084,3);P(1085,1)     P(1085,2)
P(1085,3);P(1086,1)     P(1086,2)     P(1086,3);P(1087,1)     P(1087,2)
P(1087,3);P(1088,1)     P(1088,2)     P(1088,3);P(1089,1)     P(1089,2)
P(1089,3);P(1090,1)     P(1090,2)     P(1090,3);P(1091,1)     P(1091,2)
P(1091,3);P(1092,1)     P(1092,2)     P(1092,3);P(1093,1)     P(1093,2)
P(1093,3);P(1094,1)     P(1094,2)     P(1094,3);P(1095,1)     P(1095,2)
P(1095,3);P(1096,1)     P(1096,2)     P(1096,3);P(1097,1)     P(1097,2)
P(1097,3);P(1098,1)     P(1098,2)     P(1098,3);P(1099,1)     P(1099,2)
P(1099,3);P(1100,1)     P(1100,2)     P(1100,3);P(1101,1)     P(1101,2)
P(1101,3);P(1102,1)     P(1102,2)     P(1102,3);P(1103,1)     P(1103,2)
P(1103,3);P(1104,1)     P(1104,2)     P(1104,3);P(1105,1)     P(1105,2)
P(1105,3);P(1106,1)     P(1106,2)     P(1106,3);P(1107,1)     P(1107,2)
P(1107,3);P(1108,1)     P(1108,2)     P(1108,3);P(1109,1)     P(1109,2)
P(1109,3);P(1110,1)     P(1110,2)     P(1110,3);P(1111,1)     P(1111,2)
P(1111,3);P(1112,1)     P(1112,2)     P(1112,3);P(1113,1)     P(1113,2)
P(1113,3);P(1114,1)     P(1114,2)     P(1114,3);P(1115,1)     P(1115,2)
P(1115,3);P(1116,1)     P(1116,2)     P(1116,3);P(1117,1)     P(1117,2)
P(1117,3);P(1118,1)     P(1118,2)     P(1118,3);P(1119,1)     P(1119,2)
P(1119,3);P(1120,1)     P(1120,2)     P(1120,3);P(1121,1)     P(1121,2)
P(1121,3);P(1122,1)     P(1122,2)     P(1122,3);P(1123,1)     P(1123,2)
P(1123,3);P(1124,1)     P(1124,2)     P(1124,3);P(1125,1)     P(1125,2)
P(1125,3);P(1126,1)     P(1126,2)     P(1126,3);P(1127,1)     P(1127,2)
P(1127,3);P(1128,1)     P(1128,2)     P(1128,3);P(1129,1)     P(1129,2)
P(1129,3);P(1130,1)     P(1130,2)     P(1130,3);P(1131,1)     P(1131,2)
P(1131,3);P(1132,1)     P(1132,2)     P(1132,3);P(1133,1)     P(1133,2)
P(1133,3);P(1134,1)     P(1134,2)     P(1134,3);P(1135,1)     P(1135,2)
P(1135,3);P(1136,1)     P(1136,2)     P(1136,3);P(1137,1)     P(1137,2)
P(1137,3);P(1138,1)     P(1138,2)     P(1138,3);P(1139,1)     P(1139,2)
P(1139,3);P(1140,1)     P(1140,2)     P(1140,3);P(1141,1)     P(1141,2)
P(1141,3);P(1142,1)     P(1142,2)     P(1142,3);P(1143,1)     P(1143,2)
P(1143,3);P(1144,1)     P(1144,2)     P(1144,3);P(1145,1)     P(1145,2)
P(1145,3);P(1146,1)     P(1146,2)     P(1146,3);P(1147,1)     P(1147,2)
P(1147,3);P(1148,1)     P(1148,2)     P(1148,3);P(1149,1)     P(1149,2)
P(1149,3);P(1150,1)     P(1150,2)     P(1150,3);P(1151,1)     P(1151,2)
P(1151,3);P(1152,1)     P(1152,2)     P(1152,3);P(1153,1)     P(1153,2)
P(1153,3);P(1154,1)     P(1154,2)     P(1154,3);P(1155,1)     P(1155,2)
P(1155,3);P(1156,1)     P(1156,2)     P(1156,3);P(1157,1)     P(1157,2)
P(1157,3);P(1158,1)     P(1158,2)     P(1158,3);P(1159,1)     P(1159,2)
P(1159,3);P(1160,1)     P(1160,2)     P(1160,3);P(1161,1)     P(1161,2)
P(1161,3);P(1162,1)     P(1162,2)     P(1162,3);P(1163,1)     P(1163,2)
P(1163,3);P(1164,1)     P(1164,2)     P(1164,3);P(1165,1)     P(1165,2)
P(1165,3);P(1166,1)     P(1166,2)     P(1166,3);P(1167,1)     P(1167,2)
P(1167,3);P(1168,1)     P(1168,2)     P(1168,3);P(1169,1)     P(1169,2)
P(1169,3);P(1170,1)     P(1170,2)     P(1170,3);P(1171,1)     P(1171,2)
P(1171,3);P(1172,1)     P(1172,2)     P(1172,3);P(1173,1)     P(1173,2)
P(1173,3);P(1174,1)     P(1174,2)     P(1174,3);P(1175,1)     P(1175,2)
P(1175,3);P(1176,1)     P(1176,2)     P(1176,3);P(1177,1)     P(1177,2)
P(1177,3);P(1178,1)     P(1178,2)     P(1178,3);P(1179,1)     P(1179,2)
P(1179,3);P(1180,1)     P(1180,2)     P(1180,3);P(1181,1)     P(1181,2)
P(1181,3);P(1182,1)     P(1182,2)     P(1182,3);P(1183,1)     P(1183,2)
P(1183,3);P(1184,1)     P(1184,2)     P(1184,3);P(1185,1)     P(1185,2)
P(1185,3);P(1186,1)     P(1186,2)     P(1186,3);P(1187,1)     P(1187,2)
```

```
P(1187,3);P(1188,1)      P(1188,2)      P(1188,3);P(1189,1)      P(1189,2)
P(1189,3);P(1190,1)      P(1190,2)      P(1190,3);P(1191,1)      P(1191,2)
P(1191,3);P(1192,1)      P(1192,2)      P(1192,3);P(1193,1)      P(1193,2)
P(1193,3);P(1194,1)      P(1194,2)      P(1194,3);P(1195,1)      P(1195,2)
P(1195,3);P(1196,1)      P(1196,2)      P(1196,3);P(1197,1)      P(1197,2)
P(1197,3);P(1198,1)      P(1198,2)      P(1198,3);P(1199,1)      P(1199,2)
P(1199,3);P(1200,1)      P(1200,2)      P(1200,3);P(1201,1)      P(1201,2)
P(1201,3);P(1202,1)      P(1202,2)      P(1202,3);P(1203,1)      P(1203,2)
P(1203,3);P(1204,1)      P(1204,2)      P(1204,3);P(1205,1)      P(1205,2)
P(1205,3);P(1206,1)      P(1206,2)      P(1206,3);P(1207,1)      P(1207,2)
P(1207,3);P(1208,1)      P(1208,2)      P(1208,3);P(1209,1)      P(1209,2)
P(1209,3);P(1210,1)      P(1210,2)      P(1210,3);P(1211,1)      P(1211,2)
P(1211,3);P(1212,1)      P(1212,2)      P(1212,3);P(1213,1)      P(1213,2)
P(1213,3);P(1214,1)      P(1214,2)      P(1214,3);P(1215,1)      P(1215,2)
P(1215,3);P(1216,1)      P(1216,2)      P(1216,3);P(1217,1)      P(1217,2)
P(1217,3);P(1218,1)      P(1218,2)      P(1218,3);P(1219,1)      P(1219,2)
P(1219,3);P(1220,1)      P(1220,2)      P(1220,3);P(1221,1)      P(1221,2)
P(1221,3);P(1222,1)      P(1222,2)      P(1222,3);P(1223,1)      P(1223,2)
P(1223,3);P(1224,1)      P(1224,2)      P(1224,3);P(1225,1)      P(1225,2)
P(1225,3);P(1226,1)      P(1226,2)      P(1226,3);P(1227,1)      P(1227,2)
P(1227,3);P(1228,1)      P(1228,2)      P(1228,3);P(1229,1)      P(1229,2)
P(1229,3);P(1230,1)      P(1230,2)      P(1230,3);P(1231,1)      P(1231,2)
P(1231,3);P(1232,1)      P(1232,2)      P(1232,3);P(1233,1)      P(1233,2)
P(1233,3);P(1234,1)      P(1234,2)      P(1234,3);P(1235,1)      P(1235,2)
P(1235,3);P(1236,1)      P(1236,2)      P(1236,3);P(1237,1)      P(1237,2)
P(1237,3);P(1238,1)      P(1238,2)      P(1238,3);P(1239,1)      P(1239,2)
P(1239,3);P(1240,1)      P(1240,2)      P(1240,3);P(1241,1)      P(1241,2)
P(1241,3);P(1242,1)      P(1242,2)      P(1242,3);P(1243,1)      P(1243,2)
P(1243,3);P(1244,1)      P(1244,2)      P(1244,3);P(1245,1)      P(1245,2)
P(1245,3);P(1246,1)      P(1246,2)      P(1246,3);P(1247,1)      P(1247,2)
P(1247,3);P(1248,1)      P(1248,2)      P(1248,3);P(1249,1)      P(1249,2)
P(1249,3);P(1250,1)      P(1250,2)      P(1250,3);P(1251,1)      P(1251,2)
P(1251,3);P(1252,1)      P(1252,2)      P(1252,3);P(1253,1)      P(1253,2)
P(1253,3);P(1254,1)      P(1254,2)      P(1254,3);P(1255,1)      P(1255,2)
P(1255,3);P(1256,1)      P(1256,2)      P(1256,3);P(1257,1)      P(1257,2)
P(1257,3);P(1258,1)      P(1258,2)      P(1258,3);P(1259,1)      P(1259,2)
P(1259,3);P(1260,1)      P(1260,2)      P(1260,3);P(1261,1)      P(1261,2)
P(1261,3);P(1262,1)      P(1262,2)      P(1262,3);P(1263,1)      P(1263,2)
P(1263,3);P(1264,1)      P(1264,2)      P(1264,3);P(1265,1)      P(1265,2)
P(1265,3);P(1266,1)      P(1266,2)      P(1266,3);P(1267,1)      P(1267,2)
P(1267,3);P(1268,1)      P(1268,2)      P(1268,3);P(1269,1)      P(1269,2)
P(1269,3);P(1270,1)      P(1270,2)      P(1270,3);P(1271,1)      P(1271,2)
P(1271,3);P(1272,1)      P(1272,2)      P(1272,3);P(1273,1)      P(1273,2)
P(1273,3);P(1274,1)      P(1274,2)      P(1274,3);P(1275,1)      P(1275,2)
P(1275,3);P(1276,1)      P(1276,2)      P(1276,3);P(1277,1)      P(1277,2)
P(1277,3);P(1278,1)      P(1278,2)      P(1278,3);P(1279,1)      P(1279,2)
P(1279,3);P(1280,1)      P(1280,2)      P(1280,3);P(1281,1)      P(1281,2)
P(1281,3)],m);
% disp('EL CODIGO ENTRANTE ES:')
% msg
code = rsenc(msg,n,k,genpoly);
%     Codigo(i,1)=code(1,1);
%     Codigo(i,2)=code(1,2);
%     Codigo(i,3)=code(1,3);
%     Codigo(i,4)=code(1,4);
%     Codigo(i,5)=code(1,5);
%     Codigo(i,6)=code(1,6);
```

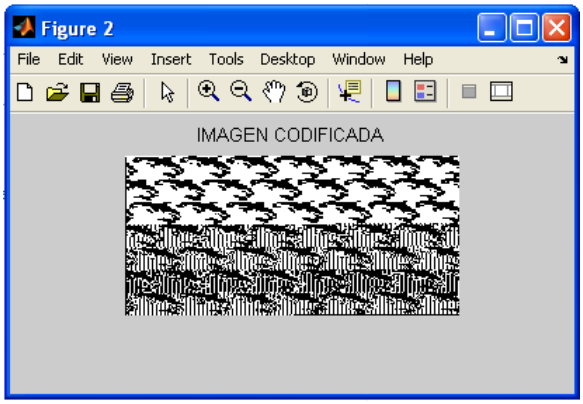
```
%     Codigo(i,7)=code(1,7);

% disp('CODIGO COMPLETO DE LA IMAGEN ORIGINAL');
% code

[codef,codec]=size(code);
for v=1:codef
    for w=1:codec
        if(code(v,w)==0)
            codeD(v,w)=0;
        elseif(code(v,w)==1)
            codeD(v,w)=1;
        elseif(code(v,w)==2)
            codeD(v,w)=2;
        elseif(code(v,w)==3)
            codeD(v,w)=3;
        elseif(code(v,w)==4)
            codeD(v,w)=4;
        elseif(code(v,w)==5)
            codeD(v,w)=5;
        elseif(code(v,w)==6)
            codeD(v,w)=6;
        elseif(code(v,w)==7)
            codeD(v,w)=7;
        end
    end
end
% disp('CODIGO COMPLETO DE LA IMAGEN ORIGINAL EN DECIMAL');
% codeD

CodBin=dec2bin(codeD,3);
CodBin
[Codf,Codc]=size(CodBin);
ctafila=1;
ctacol=0;
for x=1:Codf
    for y=1:Codc
        if(y+ctacol==239)
            ctafila=ctafila+1;
            ctacol=0;
        end
        IC(ctafila,y+ctacol)=CodBin(x,y);
    end
    ctacol=ctacol+3;
end
IC
[ICF,ICC]=size(IC);
for x=1:ICF
    for y=1:ICC
        if(IC(x,y)=='1')
            ICD(x,y)=1;
        else
            ICD(x,y)=0;
        end
    end
end
```

```
ICD
figure
imshow(ICD), title('IMAGEN CODIFICADA');
%%
%% Decodificador REED-SOLOMON (7,3)
% Decoded = rsdec(Codigo,n,k);
% Decoded
```

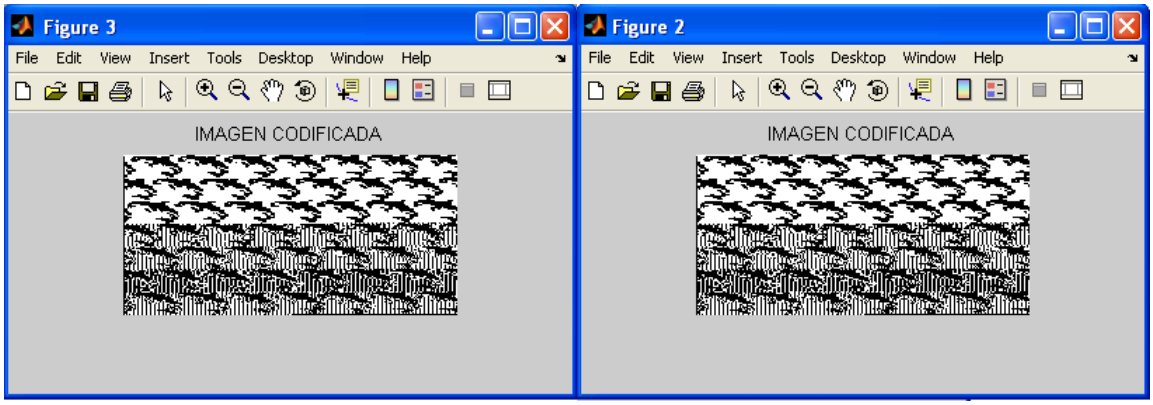
Como pueden observar fue mas sencillo con mi desarrollo de programa que utilizando las funciones de MATLAB, ya que pude observar con éxito mis resultados.



Esta fue la imagen resultante de las funciones de MATLAB.



Poniendo las 2 imágenes, una debajo de la otra se observa la similitud.



ANEXO D: PROGRAMACION EN VHDL

Por último, hice la programación y codificación para código VHDL, donde se montaría el programa para trasladarlo a la FPGA. El programa implementado es el siguiente:

```
----------------------------------------------------------------------------------
-- Company:                     INSTITUTO POLITECNICO NACIONAL
-- Engineer:                    INGENIERIA EN COMUNICACIONES Y ELECTRONICA
--
-- Create Date:         16:14:53 04/19/2010
-- Design Name:                 PRUEBA REED-SOLOMON
-- Module Name:         pruebe_rsenc - Behavioral
-- Project Name:                CODIFICADOR REE-SOLOMON
-- Target Devices:      SPARTAN 3(XC3S200-4FT256)
-- Tool versions:       10.1
-- Description:                 IMPLEMENTACION DEL CODIFICADOR REED-SOLOMON
--
-- Dependencies:                ESIME
--
-- Revision:                    "TODOS LOS DIAS"
-- Revision 0.01 - File Created
-- Additional Comments: ASESORES:
--                      M. EN C. GABRIELA SANCHEZ MELENDEZ
--                      M. EN C. DAVID VAZQUEZ ALVAREZ
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity code_rs is
        Port (   Entrada1 : in  STD_LOGIC_VECTOR (0 to 2);
                                Entrada2 : in  STD_LOGIC_VECTOR (0 to 2);
                                Entrada3 : in  STD_LOGIC_VECTOR (0 to 2);
                                Salida1 : out  STD_LOGIC_VECTOR (0 to 2);
                                Salida2 : out  STD_LOGIC_VECTOR (0 to 2);
                                Salida3 : out  STD_LOGIC_VECTOR (0 to 2);
                                Salida4 : out  STD_LOGIC_VECTOR (0 to 2);
                                Salida5 : out  STD_LOGIC_VECTOR (0 to 2);
                                Salida6 : out  STD_LOGIC_VECTOR (0 to 2);
                                Salida7 : out  STD_LOGIC_VECTOR (0 to 2));
end code_rs;

architecture Behavioral of code_rs is
```

```
--Declaracion de vector de variables para manejar loop's (SIGNAL's)
	signal dato11: std_logic_vector (0 to 2);
	signal dato21: std_logic_vector (0 to 2);
	signal dato31: std_logic_vector (0 to 2);
	signal dato41: std_logic_vector (0 to 2);
	signal dato12: std_logic_vector (0 to 2);
	signal dato22: std_logic_vector (0 to 2);
	signal dato32: std_logic_vector (0 to 2);
	signal dato42: std_logic_vector (0 to 2);
	signal dato13: std_logic_vector (0 to 2);
	signal dato23: std_logic_vector (0 to 2);
	signal dato33: std_logic_vector (0 to 2);
	signal dato43: std_logic_vector (0 to 2);

	signal D_dato1: std_logic_vector (0 to 2);
	signal D_dato2: std_logic_vector (0 to 2);
	signal D_dato3: std_logic_vector (0 to 2);

	signal memory11: std_logic_vector (0 to 2);
	signal memory21: std_logic_vector (0 to 2);
	signal memory31: std_logic_vector (0 to 2);
	signal memory41: std_logic_vector (0 to 2);
	signal memory12: std_logic_vector (0 to 2);
	signal memory22: std_logic_vector (0 to 2);
	signal memory32: std_logic_vector (0 to 2);
	signal memory42: std_logic_vector (0 to 2);
	signal memory13: std_logic_vector (0 to 2);
	signal memory23: std_logic_vector (0 to 2);
	signal memory33: std_logic_vector (0 to 2);
	signal memory43: std_logic_vector (0 to 2);

	signal rdato11: std_logic_vector (0 to 2);
	signal rdato21: std_logic_vector (0 to 2);
	signal rdato31: std_logic_vector (0 to 2);
	signal rdato41: std_logic_vector (0 to 2);
	signal rdato12: std_logic_vector (0 to 2);
	signal rdato22: std_logic_vector (0 to 2);
	signal rdato32: std_logic_vector (0 to 2);
	signal rdato42: std_logic_vector (0 to 2);


	begin
		--Declaracion de vectores de variables
		D_dato1 <= Entrada1;

		WITH D_dato1 select
		dato11<= "000" when "000", --4x0=0
					"100" when "001", --4x1=4
					"011" when "010", --4x2=3
					"111" when "011", --4x3=7
```

```
                                "110" when "100", --4x4=6
                                "010" when "101", --4x5=2
                                "101" when "110", --4x6=5
                                "001" when others;--4x7=1

        with D_dato1 select
        dato21<= "000" when "000", --7x0=0
                                "111" when "001", --7x1=7
                                "101" when "010", --7x2=5
                                "010" when "011", --7x3=2
                                "001" when "100", --7x4=1
                                "110" when "101", --7x5=6
                                "100" when "110", --7x6=4
                                "011" when others;--7x7=3

        with D_dato1 select
        dato31<= "000" when "000", --7x0=0
                                "111" when "001", --7x1=7
                                "101" when "010", --7x2=5
                                "010" when "011", --7x3=2
                                "001" when "100", --7x4=1
                                "110" when "101", --7x5=6
                                "100" when "110", --7x6=4
                                "011" when others;--7x7=3

        with D_dato1 select
        dato41<=        "000" when "000", --5x0=0
                                "101" when "001", --5x1=5
                                "001" when "010", --5x2=1
                                "100" when "011", --5x3=4
                                "010" when "100", --5x4=2
                                "111" when "101", --5x5=7
                                "011" when "110", --5x6=3
                                "110" when others;--5x7=6

memory41      <=      dato41;
memory31      <=      dato31;
memory21      <=      dato21;
memory11      <=      dato11;

rdato41 <=      memory41;
rdato31 <=      memory31;
rdato21 <=      memory21;
rdato11 <=      memory11;

D_dato2 <= rdato11 xor Entrada2;

        WITH D_dato2 select
        dato12<= "000" when "000", --4x0=0
                                "100" when "001", --4x1=4
                                "011" when "010", --4x2=3
```

```
                        "111" when "011", --4x3=7
                        "110" when "100", --4x4=6
                        "010" when "101", --4x5=2
                        "101" when "110", --4x6=5
                        "001" when others;--4x7=1

with D_dato2 select
dato22<= "000" when "000", --7x0=0
                        "111" when "001", --7x1=7
                        "101" when "010", --7x2=5
                        "010" when "011", --7x3=2
                        "001" when "100", --7x4=1
                        "110" when "101", --7x5=6
                        "100" when "110", --7x6=4
                        "011" when others;--7x7=3

with D_dato2 select
dato32<= "000" when "000", --7x0=0
                        "111" when "001", --7x1=7
                        "101" when "010", --7x2=5
                        "010" when "011", --7x3=2
                        "001" when "100", --7x4=1
                        "110" when "101", --7x5=6
                        "100" when "110", --7x6=4
                        "011" when others;--7x7=3

with D_dato2 select
dato42<=    "000" when "000", --5x0=0
                        "101" when "001", --5x1=5
                        "001" when "010", --5x2=1
                        "100" when "011", --5x3=4
                        "010" when "100", --5x4=2
                        "111" when "101", --5x5=7
                        "011" when "110", --5x6=3
                        "110" when others;--5x7=6

memory42    <=    dato42;
memory32    <=    rdato41 xor dato32;
memory22    <=    rdato31 xor dato22;
memory12    <=    rdato21 xor dato12;

rdato42 <=    memory42;
rdato32 <=    memory32;
rdato22 <=    memory22;
rdato12 <=    memory12;

D_dato3 <= rdato12 xor Entrada3;

WITH D_dato3 select
dato13<= "000" when "000", --4x0=0
                        "100" when "001", --4x1=4
```

```
                                    "011" when "010", --4x2=3
                                    "111" when "011", --4x3=7
                                    "110" when "100", --4x4=6
                                    "010" when "101", --4x5=2
                                    "101" when "110", --4x6=5
                                    "001" when others;--4x7=1
                with D_dato3 select
                dato23<= "000" when "000", --7x0=0
                                    "111" when "001", --7x1=7
                                    "101" when "010", --7x2=5
                                    "010" when "011", --7x3=2
                                    "001" when "100", --7x4=1
                                    "110" when "101", --7x5=6
                                    "100" when "110", --7x6=4
                                    "011" when others;--7x7=3
                with D_dato3 select
                dato33<= "000" when "000", --7x0=0
                                    "111" when "001", --7x1=7
                                    "101" when "010", --7x2=5
                                    "010" when "011", --7x3=2
                                    "001" when "100", --7x4=1
                                    "110" when "101", --7x5=6
                                    "100" when "110", --7x6=4
                                    "011" when others;--7x7=3
                with D_dato3 select
                dato43<=    "000" when "000", --5x0=0
                                    "101" when "001", --5x1=5
                                    "001" when "010", --5x2=1
                                    "100" when "011", --5x3=4
                                    "010" when "100", --5x4=2
                                    "111" when "101", --5x5=7
                                    "011" when "110", --5x6=3
                                    "110" when others;--5x7=6

                memory43    <=      dato43;
                memory33    <=      rdato42 xor dato33;
                memory23    <=      rdato32 xor dato23;
                memory13    <=      rdato22 xor dato13;

        Salida1     <=      Entrada1;
        Salida2     <=      Entrada2;
        Salida3     <=      Entrada3;
        Salida4     <=      memory13;
        Salida5     <=      memory23;
        Salida6     <=      memory33;
        Salida7     <=      memory43;

end Behavioral;
```

De esta forma se obtuvieron los resultados para mostrarlos en el prototipo de hardware (Mostrado en el capítulo 4).

ANEXO E: ARTICULOS PRESENTADOS EN CONGRESOS

METODOLOGÍA DE DISEÑO DE UN CODIFICADOR REED-SOLOMON (7, 3) EN FPGA

CM-18
PON 60

David Vázquez Álvarez, Jesús Espitia Juárez, Gabriela Sánchez Meléndez
Departamento de Ingeniería en Comunicaciones y Electrónica
ESIME Zacatenco I.P.N., D.F., México. Teléfono: (55) 5729 – 6000 Ext.: 54553
E-mail: dvazqueza@ipn.mx, jespitia_nastysk@hotmail.com, gsanchezgsm@yahoo.mx

Resumen: **En este artículo se presenta una recopilación de las bases teóricas empleadas para diseñar bloques funcionales del codificador Reed-Solomon y una metodología de diseño orientada a tecnología en un arreglo de compuertas programable (Field Programmable Gate Array, FPGA). Inicialmente se presenta su aplicación en DVB; el diseño del algoritmo del codificador, posteriormente se concibe la arquitectura y se captura el diseño de hardware mediante el empleo de VHDL y la herramienta de Xilinx ISE 10.1. Finalmente se revisan los resultados mediante simulaciones e implementación en la FPGA de los módulos.**

Abstract: **This paper presents a compilation of the theoretical basis used to design functional blocks of the Reed-Solomon encoder and a design methodology aimed at technology in a programmable gate array (Field Programmable Gate Array, FPGA). Initially present its application in DVB, then the encoder algorithm design, architecture is conceived later and capture hardware design by using VHDL and the tool syntax Xilinx ISE 10.1. Finally we review the results through simulations and implementation on the FPGA module.**

Palabras Claves—**Son cerca de cinco palabras o frases en orden alfabético, separados por comas.**

I. INTRODUCCIÓN

El DVB es una organización que promueve estándares aceptados internacionalmente de televisión digital, en especial para HDTV (High Definition TV) y televisión vía satelital, así como para comunicaciones de datos vía satélite.

El DVB permite ejecutar estrategias de inclusión social y reducir la brecha digital. El DVB es más que alta definición; ofrece soluciones adaptables para cada país y para todos los grupos sociales.

ROC&C'2010 – CM-18 PONENCIA RECOMENDADA POR EL **COMITÉ DE COMUNICACIONES** DEL **IEEE SECCIÓN MÉXICO** Y PRESENTADA EN LA **REUNIÓN DE OTOÑO, ROC&C'2010**, ACAPULCO, GRO., DEL 28 DE NOVIEMBRE AL 4 DE DICIEMBRE DEL 2010.

Se aprecia que la televisión digital será la única opción de accesos a los servicios de la información para una gran parte de la ciudadanía [2].

Hay una variedad muy grande de estos esquemas de modulación en uso y su variedad depende principalmente de tres factores:

- El medio de transmisión:
  - ✓ Satélite.
  - ✓ Cable.
  - ✓ Terrestre.
- La aplicación o servicio.
- El país de implantación [3].

Una de las primeras decisiones del DVB fue utilizar el MPEG-2 como estándar de compresión de vídeo y audio. Por otra parte el DVB definió las técnicas de modulación y métodos de codificación para la corrección de errores, que permitan la transmisión vía satélite, cable y terrestre (DVB-S, DVB-C y DVB-T).

Tanto el DVB estándar como los DVB de diferente transmisión, se compone de cinco elementos: la codificación fuente, codificador Reed-Solomon, un entrelazado, codificación convolucional, un filtro y un modulador. En la siguiente figura muestra el esquema y características de los elementos de los diferentes DVB.

Estándares DVB	Codificación de la fuente	Codificador Reed-Solomon	Entrelazado	Codificador Convolucional	Filtrado	Modulación
DVB-S	MPEG-2	(204,188)	12 bytes	½	Nyquist	QPSK
DVB-C	MPEG-2	(204,188)	12 bytes	½	Nyquist	QAM
DVB-T	MPEG-2	(204,188)	12 bytes	½	Nyquist	OFDM

Fig. 1.1.

El codificador Reed-Solomon empleado en los estándares DVB es el RS (204,188). Para desarrollar la metodología a emplear en el diseño de codificadores RS, se optó por trabajar con el codificador RS (7,3) y así implementarlo en la FPGA y comprobar su funcionamiento.

II. CÓDIGO REED-SOLOMON

El código fue inventado por Irving S. Reed y Gustave Solomon en el año 1960. A este código se la encuentra, actualmente, aplicación en áreas como los CD's, telefonía móvil y sondas especiales. Una de las áreas donde destaca con mayor importancia es en las comunicaciones por satélite, así como los sistemas xDLS de comunicaciones por cable, tal como se menciono anteriormente.

Los códigos cíclicos son una subclase de los códigos de bloque estándar de detección y corrección de errores que protege la información contra errores en los datos transmitidos sobre un canal de comunicaciones. Este tipo de código pertenece a la categoría FEC (Forward Error Correction), es decir, corrige los datos alterados en el receptor y para ello utiliza unos bits adicionales que permiten esta recuperación [4].

En vista de la creciente tendencia hacia el uso de dispositivos de lógica reconfigurable a alta escala de integración y de los beneficios que esta tecnología ofrece a los diseñadores de sistemas digitales mediante el empleo de un lenguaje de descripción de hardware como VHDL, que permite configurar sistemas digitales según las especificaciones demandadas por los usuarios, ajustar cambios en la programación y optimizar los diseños tratándolos en forma modular, se plantea el diseño de estos módulos de codificación bajo esta tecnología.

A. Propiedades de los códigos Reed−Solomon

El código Reed-Solomon es un subconjunto de los códigos BCH (Bose Chaudhuri Hocquenqhem), códigos cíclicos que presentan entre sus parámetros (n,k,t) una relación entre los símbolos de datos (k), del código total (n) y del número máximo de errores por ser corregidos (t), y son de bloques lineales.

Un código Reed-Solomon se especifica como RS(n,k) con símbolos de s bits. Lo anterior significa que el codificador toma k símbolos de los s bits y añade símbolos de paridad para hacer una palabra de código de n símbolos.

Existen n-k símbolos de paridad de s bits cada uno. Un decodificador puede corregir hasta t símbolos que contienen errores en una palabra de código, donde 2t = (n-k).

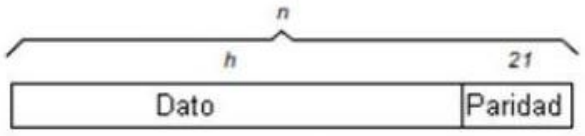


Fig. 2.1.

La Figura 2.1 muestra una típica palabra de código Reed-Solomon que se conoce como un código sistemático puesto que los datos se dejan inalterados y los símbolos de paridad se anexan.

Para codificar la trama con esta estructura se debe procesar a través de un circuito digital que opere bajo los fundamentos de campo finito de Galois. Este presenta una arquitectura en el codificador compuesta por los bloques funcionales mostrados en la Figura 2.2 [5].

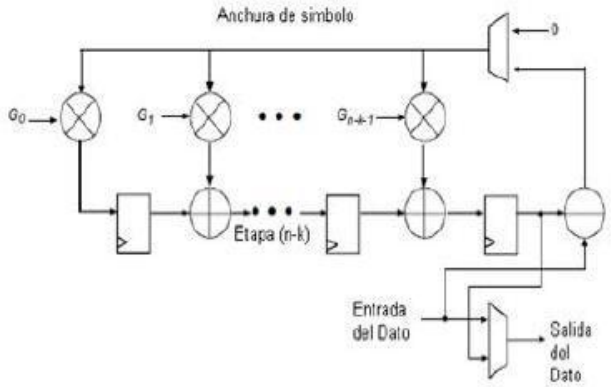


Fig. 2.2.

III. RADIO DEFINIDO POR SOFTWARE

El termino radio lo utilizamos a la combinación de transmisor (Tx) y receptor (Rx), o lo que generalmente se conoce como transceptor. Un radio es utilizado para el intercambio de información por medio de ondas electromagnéticas, y no solo puede transportar audio, sino eventualmente video y transmisiones multimedia [6].

Radio definido por software es un aspecto del dominio de procesamiento digital de señales donde señales de radio son digitalizadas y procesadas para obtener modulación o demodulación de los distintos modos.

La implementación de SDR tiene tres componentes:

1. Un dispositivo de hardware que toma la señal desde su fuente y la deja en un rango de frecuencia y amplitudes que el digitalizador puede manejar.
2. Un dispositivo que es capaz de tomar la señal y digitalizarla a una determinada velocidad máxima de muestreo, lo que a su vez define el ancho de banda máximo de la señal que es capaz de procesar.
3. Un software que es capaz de procesar la señal una vez que la misma fue digitalizada para operar sobre ella para obtener la señal demodulada final que se desea [7].

Esto se logra utilizando los convertidores analógico-digital y digital-analógico, para tomar completa ventaja del procesamiento digital de señales; los SDR's procesan las

señales digitales, para operar tantas cadenas de señales sea posible, y reconfigurarlas cerca de la antena [8].

IV. IMPLEMENTACIÓN EN HARDWARE

A partir del desarrollo teórico anterior se seleccionó el código RS (7,3), ya que es más sencillo de manipular para hacer pruebas; en vista de que logra las ventajas de corrección de errores con una aplicación práctica para programar en VHDL.

El hardware y software utilizado; en primera instancia fue desarrollado en C++ donde se realizó el algoritmo del funcionamiento del codificador Reed-Solomon utilizando los elementos requeridos tales como los campos de Galois y las compuertas x-or empleadas; que permitiera introducir y generar los datos necesarios para verificar y comprobar los resultados que se obtendrían en la pruebas de MATLAB y de testbench del ISE 10.1 de Xilinx derivados de la programación en VHDL.

En segunda instancia se desarrollo en ISE 10.1 System Generator donde se realizó el ensamblado del codificador, primeramente tomamos un generador de Bernoulli, este introducirá la información que pasará por el codificador Reed-Solomon (data_in) y así obtener el código (data_out). El codificador Reed-Solomon requiere de conexiones tales como el Bypass y el start; el Bypass se mantiene en cero lógico para que el codificador este en funcionamiento y el start es el tren de pulsos que regirá la información en un ancho de símbolo de 3; es decir, por cada pulso introducido en el start el codificador trabajará con 3 símbolos y saldrán 7 símbolos.

El pin de salida info manda el tren de pulsos para su retroalimentación; el inconveniente es que manda los pulsos inversos. Para solucionarlo se pone un inversor con retardo para controlar los pulsos introducidos en el start

Todos los pasos intermedios para la aplicación de FPGA, incluyendo la síntesis y el lugar y la ruta, se realizan casi automáticamente para generar un archivo de programación.

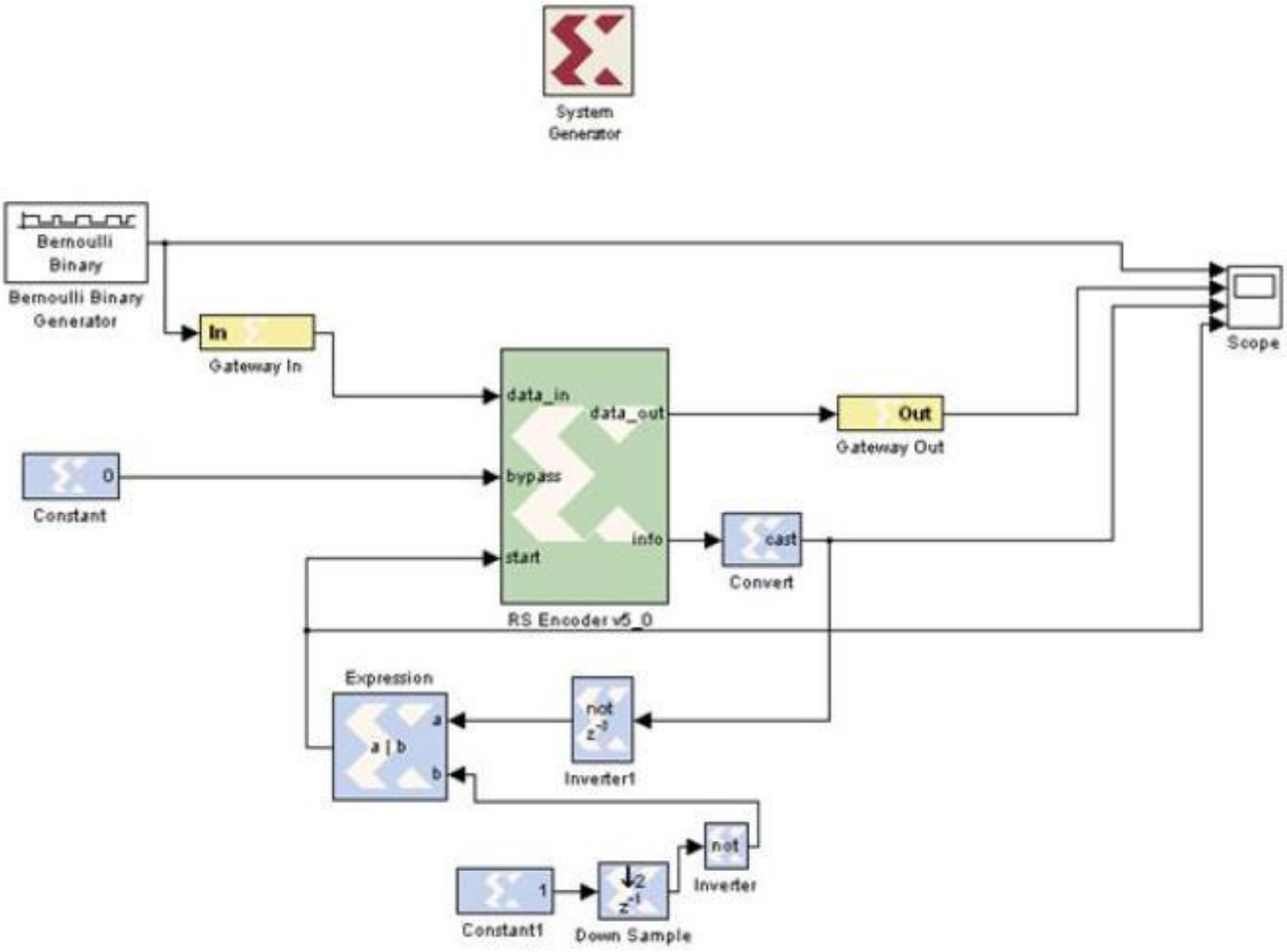


Fig. 4.1.

En la Figura 4.1 se muestran los bloques necesarios para realizar el codificador Reed-Solomon desarrollado en el Simulink de Matlab con las herramientas de Xilinx

Un elemento importante que debemos de tomar en cuenta es introducir el bloque de System Generator, ya que, proporciona el control del sistema y los parámetros de la simulación y programación de nuestra FPGA, además de ser utilizado para proporcionar el código generador para implementar nuestro programa en la FPGA.

Cada bloque puede ser modificado para una función específica según los parámetros requeridos.

El resultado de nuestra simulación del codificador Reed-Solomon (7,3) es la siguiente.

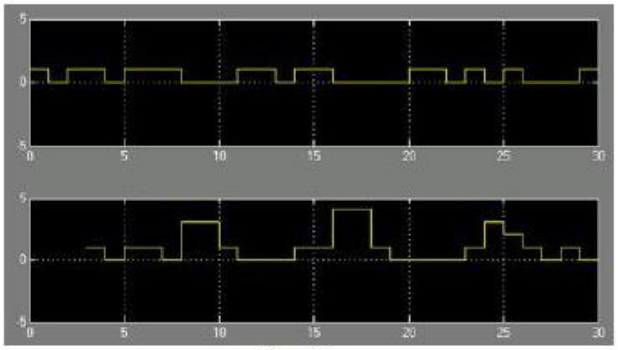


Fig. 4.2.

Cabe mencionar que no se generó el código VHDL por el método de simulación, debido a que no se contaba con los recursos y requisitos necesarios, ya que, Xilinx solicita comprar licencias de ciertos módulos para obtener los archivos completos y crear el código VHDL de los bloques empleados, tales como el codificador Reed-Solomon. Por lo tanto, desarrollamos el código VHDL de nuestro codificador Reed-Solomon (7,3).

El ISE 10.1 de Xilinx contiene un asistente para empezar a crear un proyecto de programación en VHDL, dando las especificaciones y parámetros propuestos en el nuevo proyecto; para ello se utilizó el mismo algoritmo que se desarrolló para la programación en C++ que se muestra en la figura 4.3.

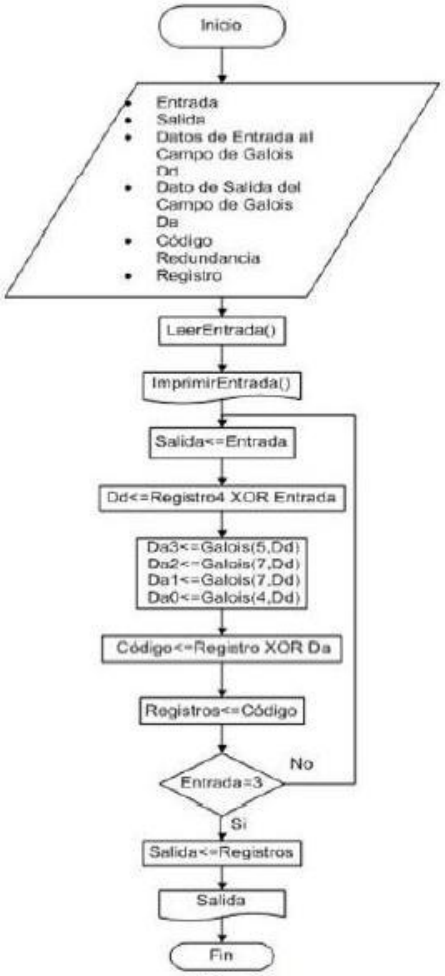


Fig. 4.3.

Para realizar las pruebas de funcionamiento de los diferentes módulos se utilizó una placa de desarrollo de la firma DIGILENT Inc. la cual monta un chip de lógica programable FPGA Spartan III XC3S200 producido por la empresa Xilinx. Dicho chip posee 200K compuertas o 4320 celdas lógicas. Para la programación, depurado (Test Bench) y carga de la programación al chip FPGA se utilizó el entorno de desarrollo ISE 10.1 de Xilinx.

En la figura 4.4 se muestran las etapas para la implementación del codificador Reed-Solomon (7,3), desde el planteamiento del objetivo, la descripción en VHDL, simulación funcional, síntesis del esquema, implementación de la programación, proceso de la grabación y el funcionamiento de la FPGA

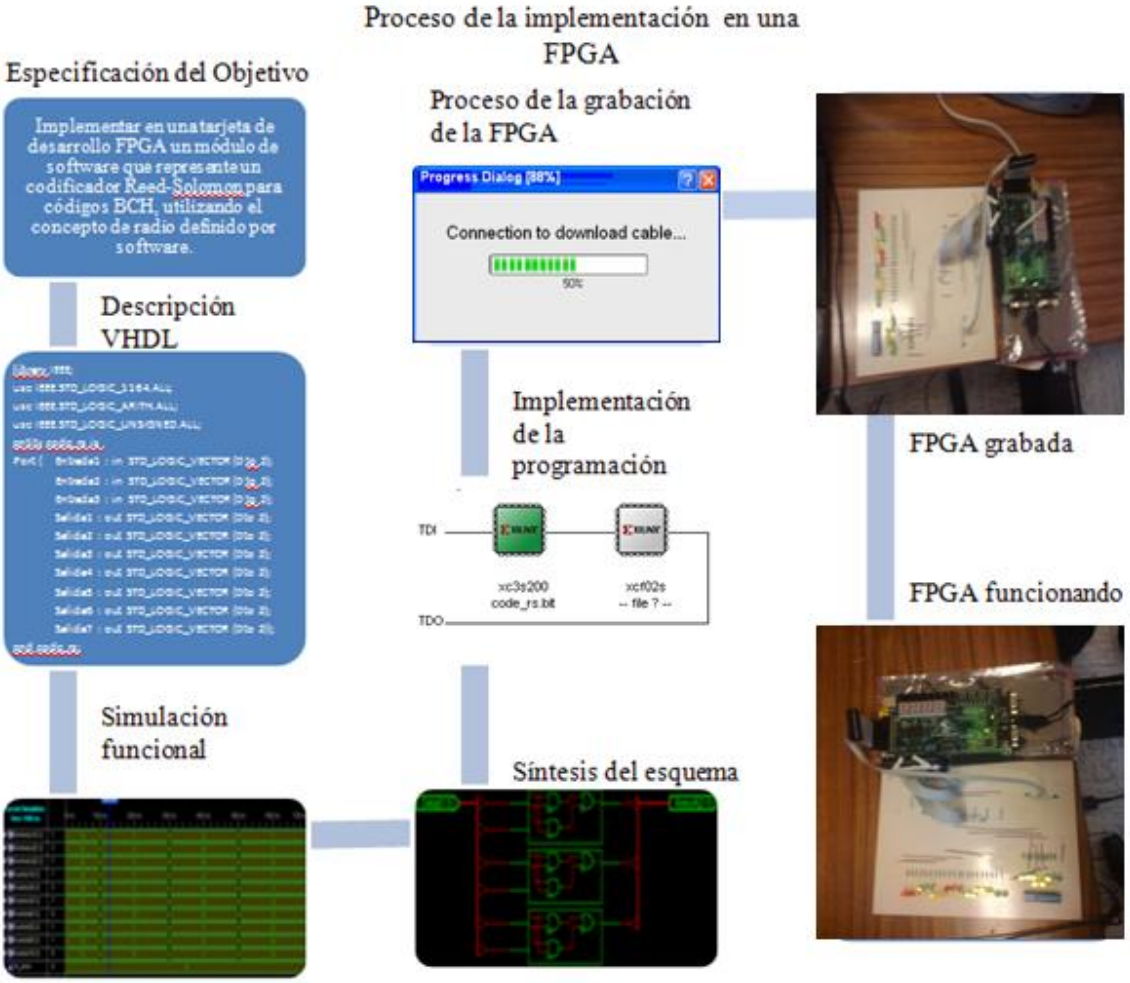


Fig. 4.4.

El resultado obtenido del codificador Reed-Solomon (7,3) en el simulador del ISE 10.1 de Xilinx se muestra en la figura 4.5.

Fig. 4.5.

V. CONCLUSIONES

El código Reed-Solomon es aplicado en comunicaciones móviles y dispositivos de almacenamiento. Se desarrolló la metodología para implementar un codificador Reed-Solomon en una FPGA para cualquier especificación de detección y corrección de errores en aplicaciones del estándar DVB, el cual está siendo uno de los elementos importantes empleados en la digitalización del video.

Con este trabajo se obtiene un modulo de software que representa a un codificador RS y que puede ser utilizado en el transmisor de cualquier categoría del estándar DVB, cabe resaltar que la metodología puede emplearse para implementar el RS (204,188) como se revisó en la Fig. 1.1. En trabajos futuros está pendiente la implementación del decodificador de Viterbi en software, para que el esquema de codificación este completo, y terminar así, transmisor y receptor DVB.

REFERENCIAS

[1] Jhonatan Peña Vazquez, David Vázquez Álvarez, Tesis de titulación "Códigos Convolucionales", I.P.N. ESIME Zacatenco, México 2009.
[2] DVB (2008). "La organización y los estándares DVB" [En linea]. Disponible: http://www.slideshare.net /Indotel/la- organizacin-dvb-los-estndares-dvb.

[3] *Sotelo, Rafael, Durán, Diego.* "Modulación Digital Aplicación a la Televisión Digital en DVB" [En linea]. Disponible: http://www.um.edu.uy/_upload/_investigación/web_investigacion_53_Memoria_5_ModulacionDigital.pdf

[4] Esperanza Romero (2007). "Código Reed-Solomon" [En linea]. Disponible: http://www.slideshare.net/bonnzai /reed-solomon-397239.

[5] Cecilia E Sandoval Ruiz, Antonio Fedon. "Codificador y decodificador digital Reed-Solomon programados para hardware Reprogramable". Universidad de Concepción, Colombia 2007.

[6] CQ Radio Amateur, Edición española, nº 243 (marzo 2004),"Radio Definido por Software" [En linea]. Disponible: http://www.ea1uro.com/sdr.html.

[7] Ing. Pedro E. Colla (2010) "Apuntes sobre Radio Definida por Software". Radio club Córdoba, Argentina [En linea]. Disponible: *www.qsl.net/l/lu7did//docs/SDR/SDR_Apuntes_v1.1.doc*.

[8] Texas instrument (28 abril 2009). "Smart Selection of ADC/DAC Enables Better Design of Software-Defined Radio" [En linea]. Disponible: http://focus.ti.com/general/docs/lit/getliterature.tsp?literatureNumber=slaa407&fileType=pdf.

BIOGRAFÍAS.

M. en C. David Vázquez Álvarez. Maestro en Ciencias en Ingeniería de Telecomunicaciones en la SEPI ESIME IPN. Actualmente es presidente de la Academia de Computación del Departamento de Ingeniería en Comunicaciones y Electrónica en la ESIME IPN. Áreas de Interés Actual: Fibras Ópticas, Radio Definido por Software, Radio Cognitivo, Sistemas de Comunicaciones Móviles e Inalámbricas.

Jesús Espitia Juárez. Alumno-Investigador en el Instituto Politécnico Nacional (ESIME Zacatenco), México. Pasante de Ingeniería en Comunicaciones y Electrónica. Actualmente se encuentra desarrollando su proyecto de tesis de titulación. Siendo su área de interés el radio definido por software.

M. en C. Gabriela Sánchez Meléndez. Maestra en Ciencias en Ingeniera de Telecomunicaciones de la Sección de Estudios de Posgrado e Investigación del IPN. Actualmente es Profesora Asociada de la Escuela Superior de Ingeniería Mecánica y Eléctrica unidad Profesional Adolfo López Mateos. Áreas de Interés: Fibras Ópticas, Telecomunicaciones,

REFERENCIAS:

[1] Component-Based Software Defined Radio and JTRS-SCA in PocoCapsule. Ke Jin, Oct 17, 2007. Copyright(c) 2007 by Pocomatic Software. All Rights Reserved.

[2] Revista CQ RadioAmateur, Edición española, nº 243 (marzo 2004), http://www.ea1uro.com/sdr.html. Fecha de revisión: junio 2009.

[3] http://www.tools4sdr.com/articles/Description+of+cognitive+radio+concept_2504610 . Fecha de revisión: junio 2009.

[4] http://focus.ti.com/docs/solution/folders/print/357.html. Fecha de revisión: Junio 2009.

[5] DVB (2008). "La organización y los estándares DVB" [En linea]. Disponible: http://www.slideshare.net /Indotel/la- organizacin-dvb-los-estndares-dvb

[6] *Sotelo, Rafael, Durán, Diego. "*Modulación Digital Aplicación a la Televisión Digital en DVB" [En linea]. Disponible: http://www.um.edu.uy/_upload/_ *investigación/web_investigacion_53_Memoria_5_ModulacionDigital.pd*

[7] http://opensdr.sourceforge.net/. Fecha de revisión: junio 2009.

[8] http://www.worldlingo.com/ma/enwiki/es/BCH_code. Fecha de revisión: noviembre 2009.

[9] http://ocw.upm.es/teoria-de-la-senal-y-comunicaciones-1/sistemas-de-telecomunicacion/Contenidos/Apuntes/6_distribucion_video_por_satelite.pdf. Fecha de revisión: noviembre 2009.

[10] http://148.206.53.231/tesiuami/uamr0082.pdf. Fecha de revisión: noviembre 2009.

[11] Leon W. Couch, "Sistemas de comunicación digitales", pag. 653, 4th edition, Prentice Hall, 2001.

[12] Artículo: **Metodología de prototipado rápido desde Matlab: Herramientas visuales para flujo de datos.** Moisés Serra (1), Oscar Navas (2), Jordi Escrig (2) , Martí Bonamusa (2) ,Pere Martí (1), Jordi Carrabina (2).
1: Dept. d'Electrónica i Telecomunicacions, Universitat de Vic.
2: Dept. Informática, ETSE, Universitat Autónoma de Barcelona.

[13] http://neo.lcc.uma.es/evirtual/cdd/tutorial/enlace/detec.html. Fecha de revisión: diciembre 2009.

[14] http://enciclopedia.us.es/index.php/Codificador. Fecha de revisión: diciembre 2009.

[15] Arriagada, Álvaro. *FEC (Forward Error Correction) y Código Reed- Solomon*. Universidad de Concepción, 2001.

[16] http://www3.euitt.upm.es/taee/congresos_taee/libros_de_actas/taee06/papers/S7/p198.pdf. Fecha de revisión: noviembre 2009.

[17] Artículo: **FPGA: Nociones básicas e implementación**. M. L. López Vallejo y J. L. Ayala Rodrigo.

[18] http://www.lionelremigio.com/que%20es%20fec.htm. Fecha de revisión: diciembre 2009.

[19] Lara Marín Vinuesa, Spectrum Management in Cognitive radio Networks, http://upcommons.upc.edu/pfc/bitstream/2099.1/4922/1/memoria.pdf. Fecha de revisión: Junio de 2009.

[20] Texas Instruments Incorporated, Printed in the U.S.A. SPRT348A, 2005, http://focus.ti.com/lit/ml/sprt348a/sprt348a.pdf. Fecha de revisión: Junio 2009.

[21] Joseph Mitola, ISBN: 047138492-5, October 2000, Wiley.

[22] Symon Haykin, “Communication Systems”, pag. 653, 4th edition, John Wiley & Sons, 2001.
[23] John G. Proakis Digital Communications. Mc Graw Hill, 2001.

[24] Xilinx Logicore, Reed-Solomon encoder, DS251 abril 28, 2005.

GLOSARIO.

ADC	(Analog to Digital Converter) Convertidor Analógico Digital, es un dispositivo electrónico que su función es convertir voltajes en una representación digital de unos y ceros, que puede ser guardada, manipulada y convertida de vuelta a analógica.
Codificar	Representar cada uno de los símbolos provenientes de una fuente por medio de un conjunto de símbolos predefinidos.
Códigos BCH	(Bose, Ray-Chaudhuri, Hocquenghem codes), códigos de corrección de errores de Bose, Ray-Chaudhuri y Hocquenghem.
Corrección de errores	Posibilidad que se tiene en las comunicaciones digitales de corregir ciertos errores que ocurran en una transmisión.
DAC	Digital to Analog Converter) Convertidor Digital Analógico, es exactamente lo opuesto que un ADC, y su función es tomar la señal digital compuesta por unos y ceros y "ensamblar" de vuelta la forma de onda analógica.
Decodificador	Dispositivo que recibe las órdenes enviadas desde la oficina central y convierte las señales digitales comprimidas en video analógico y en audio. El Set-Top Box o decodificador permite decodificar las señales de televisión transportadas por la red de cable. En un extremo el decodificador se conecta a la toma de usuario y por el otro al televisor o al aparato de vídeo.
Detección	Es el proceso de decidir cuál de las posibles señales que puede originar una fuente es la que con mayor probabilidad generó una señal recibida
Detección de errores	Es la posibilidad que se tiene en las comunicaciones digitales de identificar la ocurrencia de ciertos errores en una transmisión.
DSP	Procesadores digitales de señal son microprocesadores específicamente diseñados para el procesado digital de señal. Estrictamente hablando, el término DSP se aplica a cualquier chip que trabaje con señales representadas de forma digital
DVB (Transmisión de Video Digital.)	Es una solución propuesta para la televisión digital y la transmisión de datos por todo el rango de medios de entrega. Todos los sistemas DVB se basan en compresión de audio y video MPEG-2. DVB agrega al flujo múltiple de transporte MPEG los elementos necesarios para llevar los servicios de transmisión digital al hogar a través del cable, el satélite y los sistemas de transmisión terrestres.
DVB-T	(Digital Video Broadcasting Terrestrial), difusión digital de vídeo terrestre.
DVB-T2	(Digital Video Broadcasting Terrestrial version 2), difusión digital de vídeo terrestre versión 2.
DVB-S:	(Digital Video Broadcasting by Satellite), difusión digital de vídeo por satélite.
GSM	El diminutivo de Sistema global de comunicaciones móviles, es la plataforma de telefonía móvil más utilizada en todo el mundo.
HDTV	(High Definition TV), Televisión de alta definición

IEEE	Corresponde a las siglas de The Institute of Electrical and Electronics Engineers, el Instituto de Ingenieros Eléctricos y Electrónicos, una asociación técnico-profesional mundial dedicada a la estandarización, entre otras cosas. Es la mayor asociación internacional sin fines de lucro formada por profesionales de las nuevas tecnologías, como ingenieros eléctricos, ingenieros en electrónica, científicos de la computación, ingenieros en informática e ingenieros en telecomunicación.
INTEROPERABILIDAD	Posibilidad de combinar conjuntos de datos espaciales y de interacción de los servicios, sin intervención manual repetitiva, de forma que el resultado sea coherente y se aumente el valor agregado de los conjuntos y servicios de datos
MPEG-2	Norma técnica internacional de compresión de imagen y sonido. El MPEG-2 especifica los formatos en que deben de representarse los datos en el descodificador y un conjunto de normas para interpretar estos datos. Es un estándar definido específicamente para la compresión de vídeo, utilizado para la transmisión de imágenes en vídeo digital. El algoritmo que utiliza además de comprimir imágenes estáticas compara los fotogramas presentes con los anteriores y los futuros para almacenar sólo las partes que cambian. La señal incluye sonido en calidad digital.
PDA	Se trata de pequeñas computadoras, con funciones similares a las de una convencional, más las de funciones de una agenda electrónica. Estos dispositivos son totalmente portátiles ya que son del tamaño de la mano y muy delgados, utilizan un sistema operativo y tienen aplicaciones específicas a nivel usuario como aplicaciones ofimáticas (procesadores de palabras, hojas electrónicas), juegos, interfaz para conexión a redes, etc.
Transceptor	Es la unión de un Receptor con un Transmisor en un solo equipo. Es un dispositivo que realiza, dentro del mismo chasis, funciones de trasmisión y recepción, utilizando componentes de circuito comunes para ambas funciones. Dado que determinados elementos se utilizan tanto para la transmisión como para la recepción, la comunicación que provee un transceptor solo puede ser semidúplex, lo que significa que pueden enviarse señales entre dos terminales en ambos sentidos, pero no simultáneamente.
Transmisión por secuencias	La transmisión por secuencias hace referencia a transmisiones de vídeo o audio digitales a través de Internet. El sonido y los datos de imágenes se transmiten al abonado como una secuencia de datos, de ahí el término “transmisión por secuencias”. La ventaja principal de la transmisión por secuencias es que los usuarios pueden ver o escuchar los soportes tal y como se reciben, sin tener que esperar a que finalice la descarga
Radio cognitiva	Es un paradigma de la comunicación inalámbrica en la cual tanto las redes como los mismos nodos inalámbricos cambian los parámetros particulares de transmisión o de recepción para ejecutar su cometido de forma eficiente sin interferir con los usuarios autorizados.
Wi-Fi	Abreviación del inglés wireless fidelity (fidelidad inalámbrica). Hace referencia a determinados tipos de redes de área local inalámbrica (WLAN) que usan especificaciones que cumplen la norma IEEE 802.11b. La tecnología Wi-Fi ha ganado aceptación en muchos entornos como alternativa a una red LAN con cables. Muchos aeropuertos, hoteles y otros servicios ofrecen acceso público a redes Wi-Fi para que la gente pueda conectarse a Internet y recibir correo electrónico en sus viajes. Estas ubicaciones se denominan hotspots.