ANÁLISIS DEL DESEMPEÑO DE UN SISTEMA DE COMUNICACIONES CON CODIFICACIÓN BCH BINARIA BASADO EN HARDWARE RECONFIGURABLE

Aura Cristina Tobar Collazos
Diana Patricia Sánchez Mulcué

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telecomunicaciones
GRIAL – Grupo de Radio e InALámbricas
GNTT – Grupo I+D Nuevas Tecnologías en Telecomunicaciones
Señales y Sistemas de Acceso y Difusión Basados en Radio
Gestión Integrada de Redes, Servicios y Arquitecturas de Telecomunicaciones
Popayán, 2015

ANÁLISIS DEL DESEMPEÑO DE UN SISTEMA DE COMUNICACIONES CON CODIFICACIÓN BCH BINARIA BASADO EN HARDWARE RECONFIGURABLE

Trabajo de Grado presentado como requisito para obtener el título de Ingeniero en Electrónica y Telecomunicaciones

Aura Cristina Tobar Collazos
Diana Patricia Sánchez Mulcué

Director: Harold Armando Romo Romero

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telecomunicaciones
GRIAL – Grupo de Radio e InALámbricas
GNTT – Grupo I+D Nuevas Tecnologías en Telecomunicaciones
Señales y Sistemas de Acceso y Difusión Basados en Radio
Gestión Integrada de Redes, Servicios y Arquitecturas de Telecomunicaciones
Popayán, 2015

AGRADECIMIENTOS

A Dios y a la Virgen María porque día a día guiaron nuestros pasos, nos llenaron de fortaleza en los momentos difíciles y nos permitieron alcanzar este nuevo logro en nuestras vidas.

A nuestra Familia por ser nuestra fuente de motivación e inspiración para poder superarnos y por apoyarnos en las decisiones que tomamos día a día.

A nuestro Director Harold Romo quien sin esperar nada a cambio compartió sus conocimientos, nos inculcó responsabilidad y rigor académico, sin los cuales no podríamos tener una formación completa como la tenemos hoy.

CONTENIDO

LISTA DE FIGURAS

Pág.

LISTA DE TABLAS

Pág.

LISTA DE ACRÓNIMOS

ARQ	*Automatic Repeat Request,* Solicitud de Repetición Automática.
ADC	*Analog to Digital Converter,* Conversor Analógico a Digital.
ASIC	*Application Specific Integrated Circuit*, Circuito Integrado de Aplicación Específica.
AWGN	*Additive White Gaussian Noise,* Ruido Gausiano Aditivo Blanco.
BCH	Bose, Chaudhuri, and Hocquenghem.
BER	*Bit Error Rate*, Tasa de Error de Bit.
CLB	*Configurable Logic Block*, Bloques Lógicos Configurables.
CPLD	*Complex Programmable Logic Device,* Dispositivo Lógico Programable Complejo.
DAC	*Digital to Analog Converter,* Conversor Digital a Analógico.
DCM	*Digital Clock Manager*, Administrador de Reloj Digital.
DDC	*Digital Down Converter,* Conversor Digital de Bajada.
DSP	*Digital Signal Processor*, Procesador Digital de Señales.
DUC	*Digital Up Converter,* Conversor Digital de Subida.
FEC	*Forward Error Correction,* Corrección de Errores Hacia Adelante.
FPGA	*Field Programmable Gate Array,* Arreglo de Compuertas Programables en Campo.
FSK	*Frequency Shift Keying,* Modulación por Desplazamiento de Frecuencia.
IDE	*Integrated Development Environment,* Ambiente de Diseño Integrado.
IF	*Intermediate Frequency,* Frecuencia Intermedia.
ISE	*Integrated Software Environment,* Ambiente de Software Integrado.
HDL	*Hardware Description Language*, Lenguaje de Descripción de Hardware.
LFSR	*Linear Feedback Shift Register,* Registro de Desplazamiento con Retroalimentación Lineal.
MSK	*Minimum-Shift Keying,* Modulación por Desplazamiento Mínimo.
RAM	*Random Access Memory,* Memoria de Acceso Aleatorio.

RF	*Radio Frequency,* Radio Frecuencia.
PROM	*Programmable Read-Only Memory*, Memoria Programable de Solo Lectura.
RS	Reed Solomon.
SDR	*Software Defined Radio*, Radio Definido por Software.
USB	*Universal Serial Bus,* Bus Serial Universal.
USRP	*Universal Software Radio Periférico,* Software Radio Periférico Universal.
VHDL	*VHSIC Hardware Description Language,* Lenguaje de Descripción Hardware para VHSIC.
VHSIC	*Very High Speed Integrated Circuit,* Circuito Integrado de Muy Alta Velocidad.
WER	*Word Error Rate,* Tasa de Error de Palabra.
XSG	Xilinx System Generator.

INTRODUCCIÓN

Los sistemas de comunicaciones digitales actualmente son muy importantes porque están inmersos en las relaciones del ser humano, permitiendo la comunicación a pequeñas y largas distancias, por tal razón es necesario explorar los aspectos positivos y negativos que estos generan en diferentes ámbitos. Como ventajas se tienen: la gran capacidad de integración, flexibilidad, escalabilidad, reducción de costos frente a los sistemas de comunicación analógica, además la robustez de la señal transmitida frente a efectos degradantes como el ruido y la interferencia. Sin embargo, existe una desventaja, debido a los costos de implementación sobre hardware, lo cual afecta de manera significativa a los entornos académicos que desean interactuar con estos sistemas y analizar su desempeño. Por lo cual, existe la necesidad de contar con dispositivos hardware, flexibles y reconfigurables que permitan disminuir los costos de implementación.

Dentro de los dispositivos hardware reconfigurables, se destacan los Arreglos de Compuertas Programables en Campo (FPGA, *Field Programmable Gate Array)* que por su capacidad y flexibilidad son utilizados en diferentes entornos. Las aplicaciones más comunes en las que se utiliza este dispositivo, son el procesamiento digital de señales, el Radio Definido por Software (SDR, *Software Defined Radio*), los sistemas aeroespaciales y de defensa, imágenes médicas, visión artificial, reconocimiento de voz, bioinformática, emulación de hardware de computador, sistemas de comunicación de datos de alta velocidad, control vehicular, sistemas de medición remotos, entre otras [1]. Es importante mencionar que su uso en otros campos es cada vez mayor, sobre todo en aquellas aplicaciones que requieren un alto grado de paralelismo[1]. En las telecomunicaciones, permite evaluar el desempeño de la codificación de fuente, de canal y los diferentes esquemas de modulación.

De lo anterior, se puede comprobar el gran potencial que brindan estos dispositivos y el nivel de recursos que tienen para diseñar e implementar sistemas de comunicaciones; por tal motivo, el presente trabajo realiza sobre el FPGA distintos códigos BCH (Bose, Chaudhuri, and Hocquenghem) binarios y compara el desempeño con los esquemas de Modulación por Desplazamiento de Frecuencia (FSK, *Frequency Shift Keying)* y Modulación por Desplazamiento Mínimo (MSK, *Minimum Shift* Keying) desarrollado en la fase 1 del proyecto “Diseño e

[1] Paralelismo: Ejecución de varios procesos al tiempo.

Implementación de un Prototipo de Comunicación de Datos Basado en Hardware Reconfigurable Fase 1" [2].

La estructura del trabajo está diseñada en cuatro capítulos. El primer capítulo presenta las bases del álgebra moderna utilizada para fundamentar los códigos BCH binarios y el proceso para construirlos. El segundo, está dedicado al diseño, simulación, implementación y validación de los sistemas de comunicación digital con codificación BCH binaria integrado con los esquemas de modulación FSK/MSK.

El tercer capítulo, está integrado por las pruebas que fueron realizadas para evaluar el desempeño de los códigos BCH binarios con los esquemas de modulación FSK/MSK, principalmente con el parámetro de la Tasa de Error de Bit (BER, *Bit Error Rate*) en función de la relación Energía de bit a Densidad Espectral de Potencia de Ruido E_b/N_o. Se especifican cuatro modelos durante el desarrollo del mismo, que son el modelo teórico, el modelo en software (Simulink), el modelo en System Generator y el modelo del FPGA (hardware).

Finalmente, el cuarto capítulo contiene las principales conclusiones obtenidas a partir de los resultados gráficos que brinda el capítulo 3 mediante una inferencia comparativa de estas, de igual forman se especifican una serie de recomendaciones y propuestas a futuros trabajos.

CAPÍTULO 1

GENERALIDADES

1.1. SISTEMA DE COMUNICACIÓN DIGITAL

Es un conjunto de componentes que permiten realizar el proceso de la comunicación; encargados de transmitir la información desde un punto denominado fuente hasta otro llamado destino; en el cual los mensajes transmitidos pertenecen a un conjunto finito y discreto de valores. En la figura 1.1 se visualiza los siete componentes básicos.

La información de la fuente es convertida en una secuencia de dígitos binarios (bits), mediante el proceso de codificación de fuente o también llamada comprensión de datos, después esta secuencia de bits pasa a través del codificador de canal la cual es transformada en una secuencia codificada discreta denominada *palabra código*, en la que se agregan bits de paridad (bits de redundancia añadidos a los datos) de forma controlada para minimizar los efectos producidos en el canal.

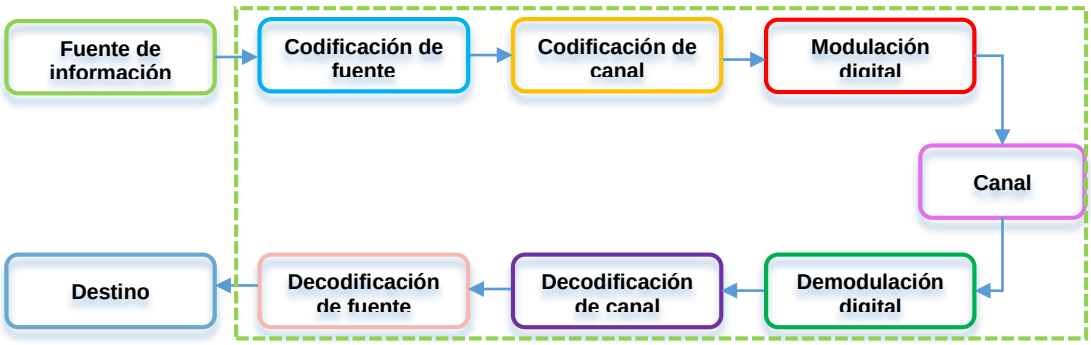


Figura 1.1 Diagrama de bloques de un sistema de comunicaciones digitales.

Posteriormente, la señal codificada es enviada al modulador digital, el cual la convierte en una forma de onda con una duración de T segundos, adecuada para ser transmitida y ser enviada al canal, que es el medio físico por donde la información viaja del transmisor al receptor; generalmente puede ser un cable de par trenzado, coaxial, fibra óptica o un enlace inalámbrico. Una característica importante es que la señal transmitida se ve afectada por diferentes fenómenos que en éste se producen, como el ruido, la interferencia, distorsión, entre otros. En el receptor, el demodulador y decodificador tanto de canal como de fuente hacen los procesos inversos al

modulador digital, codificador de canal y de fuente respectivamente, convirtiendo la señal a su forma original para llegar al destino [3].

1.1.1. Codificación de canal.

En la teoría de la información se tratan dos problemas, uno relacionado con la eficiencia y otro con la fiabilidad. El primero está relacionado con la codificación de fuente y el segundo con la de canal, que hace referencia a la forma de procesar la información en presencia de errores cuando se transmite el mensaje.

En la codificación de canal, existen dos estrategias para el control de errores, una de ellas es la Solicitud de Repetición Automática (ARQ, *Automatic Repeat Request),* en la cual, el receptor, solicita al transmisor que envié nuevamente la información, por lo cual los sistemas de comunicación que utilizan esta técnica requieren un enlace semidúplex[2]. En los casos en que no es posible la retransmisión, la técnica utilizada es la Corrección de Errores Hacia Adelante (FEC, *Forward Error Correction*), en la que los bits redundantes agregados a la información permiten detectar y corregir la mayor cantidad de errores posibles.

Dentro de los codificadores FEC, de acuerdo a como se introduce la redundancia se identifican dos clases: el primero son los códigos convolucionales, donde los bits de paridad se agregan de forma continua a medida que llega la información al codificador y el segundo, los códigos bloque (n, k), en el cual se generan n bits de código por k bits de información, adicionando una redundancia de $n-k$ bits $(n > k)$ [4]. En este último grupo se encuentra la codificación BCH de tipo binaria y no binaria que se profundiza en la sección 1.3.

Existen varios criterios, que se deben tener en cuenta para realizar una codificación fiable, los cuales se enuncian a continuación:

- **Teorema de Shannon-Codificación de Canal**

Si la tasa de transmisión R_s es menor a la capacidad del canal C, $R_s < C$, existe una técnica de codificación de canal que permite la transmisión con una BER[3] pequeña. Por el contrario, sí $R_s > C$ la probabilidad de error puede llegar a ser muy alta y no es posible reconstruir el mensaje en la salida del canal.

[2] Enlace semidúplex: Las trasmisiones se realizan entre dos terminales en ambas direcciones, pero la forma no es simultánea.

[3] BER: Parámetro para medir el desempeño del sistema. Representa el número de bits errados respecto a los transmitidos, medidos en recepción en un intervalo de tiempo dado.

El término capacidad de canal C, hace referencia a la cantidad máxima de información que puede transferir confiablemente el canal con una baja probabilidad de error [3].

- **Tasa de Codificación**

La tasa de codificación R representa la relación entre el número de bits que entran al codificador (k) y el número de bits que se obtiene a la salida de éste (n) dada por la ecuación 1.1 [3].

$$R = \frac{k}{n}. \tag{1.1}$$

- **Ganancia de Codificación**

Es la cuantificación de la mejora de un código de canal, define la reducción (expresada en dB) de la relación Energía de bit a Ruido del canal Normalizado E_b/N_0 necesaria para obtener una determinada BER, en un sistema codificado frente a otro sin codificar, como se indica en la ecuación 1.2 [5].

$$G[dB] = \frac{E_b}{N_0}u - \frac{E_b}{N_0}\text{c}. \tag{1.2}$$

- **Distancia Mínima**

Se define como el menor peso Hamming[4] de las 2^k posibles palabras código (sin incluir al vector 0) y se denota como $d_{mín}$ [6].

- **Capacidad de Corrección**

La capacidad de corrección t de un código BCH binario, se define como el número máximo de bits errados que el decodificador puede corregir. Este valor depende de la $d_{mín}$ del código (n, k) y se expresa según la ecuación 1.3 [3].

$$t = \left\lfloor \frac{d_{mín} - 1}{2} \right\rfloor. \tag{1.3}$$

1.2. FUNDAMENTOS DE LOS CÓDIGOS BCH

Los sistemas algebraicos son estructuras que satisfacen ciertas reglas o leyes, las cuales habitualmente son las mismas que se aplican en los sistemas de numeración

[4] Peso Hamming: Se define como el número de "$1s$" en una palabra código.

ordinarios. Dichas estructuras son deseables en los códigos correctores de error por dos razones: facilitan la búsqueda de propiedades de un código y llevan a cabo la implementación de códigos prácticos. Teniendo en cuenta la teoría, la codificación se suele trabajar sobre conjuntos cuyo alfabeto es bastante restringido. Es aquí en que el álgebra abstracta[5] y la aritmética modular apoyan las aplicaciones de los sistemas de comunicaciones [3]. Dentro de la primera se definen los siguientes componentes:

- **Grupo:** Es un sistema con una operación y su inversa respectiva; puede ser la adición o la multiplicación.

- **Anillo:** Tiene dos operaciones, adición y multiplicación y la operación inversa de la primera que corresponde a la substracción.

- **Campo:** Tiene las dos operaciones básicas anteriores, ambas con sus operaciones inversas cuyas propiedades son:

  - ✓ Constituye un grupo conmutativo bajo la adición, su elemento identidad, llamado elemento cero es la identidad aditiva denotada por el “0”.
  - ✓ El conjunto de elementos no nulos componen un grupo conmutativo bajo la multiplicación, su elemento identidad es la identidad multiplicativa denotada por el “1”.
  - ✓ La multiplicación es distributiva bajo la adición.

1.2.1. Campos de Galois

Los campos finitos o campos de Galois (nombre dado en honor al matemático francés Évariste Galois), proporcionan las bases matemáticas para la codificación de datos tanto para la corrección de errores como para el cifrado de datos [7]. La denotación para el campo de Galois de orden p[6] está dada por su abreviatura en inglés “Galois Field” $GF\ (p)$ o Fp; donde p debe ser un número primo, propiedad fundamental que se debe garantizar.

Para hacer uso de los campos de Galois se deben tener en cuenta los siguientes conceptos:

[5] Álgebra abstracta: Es la parte de las matemáticas que estudia las estructuras algebraicas como las de grupo, anillo, cuerpo o espacio vectorial.

[6] Orden p: Define el número total de elementos que constituyen un campo.

- **Operación módulo-n**

Dados cualquier par de enteros n y a, al dividir a entre n hay un residuo entero r y un cociente q, como se define en la ecuación 1.4.

$$a = qn + r, \tag{1.4}$$

de esta forma se define el módulo n de a ($a\ mod\ n$), como el residuo de dividir a entre n. Por ejemplo, con $a = 8$ y $n = 5$.

$$8 = (1x5) + 3,$$

donde $q = 1$ y $r = 3$, de tal forma que:

$$8 \bmod 5 = 3.$$

- **Aritmética modular**

Dado un entero positivo p, Si Z_p denota al conjunto de todos los enteros positivos módulo p:

$$Z_p = \{0, 1, 2, \dots, p-1\},$$

la aritmética modular, especifica las operaciones de adición y multiplicación dentro de un campo como se enuncia a continuación:

Sean $x, y \in Z_p$. La suma de x y y en Z_p es el residuo que resulta de dividir $x + y \in Z_p$ entre p, como se indica en la ecuación 1.5.

$$x + y = res\left\{\frac{x+y}{p}\right\}, \tag{1.5}$$

análogamente el producto de x y y en Z_p, es el residuo que resulta de dividir $x \cdot y \in Z_p$ entre p, indicado en la ecuación 1.6.

$$x \cdot y = res\left\{\frac{x \cdot y}{p}\right\}. \tag{1.6}$$

1.2.2. Campo binario

El campo binario de orden $p = 2$, está dado por los elementos $GF(2) = \{0,\ 1\}$, donde se establecen las operaciones de adición (XOR) y multiplicación (AND) indicadas en la tabla 1.1.

Tabla 1.1 Operaciones módulo-2.

Adición módulo-2	Multiplicación módulo-2
$\mathbf{0 \oplus 0 = 0}$	$0 \cdot 0 = 0$
$\mathbf{0 \oplus 1 = 1}$	$0 \cdot 1 = 0$
$\mathbf{1 \oplus 0 = 1}$	$1 \cdot 0 = 0$
$\mathbf{1 \oplus 1 = 0}$	$1 \cdot 1 = 1$

1.2.3. Polinomios en el campo $GF(2)$

En el campo $GF(2)$, existen polinomios $f(x)$ de grado n con variable x, coeficientes f_n provenientes del campo binario $\{0,1\}$. En general, hay 2^n polinomios en el campo $GF(2)$ con grado n descritos por la ecuación 1.7.

$$f(x) = f_0 + f_1 x + f_2 x^2 + \ldots + f_n x^n. \tag{1.7}$$

- **Polinomio irreducible en el campo $GF(2)$**

Un polinomio $f(x)$, con coeficientes de $GF(2)$ y grado n, se dice que es irreducible si no puede ser factorizado en polinomios de grado menor que n y mayor que 0. En la tabla A.1 del anexo A, se da un ejemplo del polinomio irreducible $f(x) = x^4 + x + 1$.

- **Polinomios primitivos**

Cualquier polinomio irreducible sobre $GF(2)$ de grado m, divide exactamente al polinomio dado por la ecuación 1.8.

$$x^{2^m - 1} + 1, \tag{1.8}$$

por ejemplo, el polinimonio irreducible $x^4 + x + 1$ de orden $m = 4$, divide exactamente al binomio $x^{15} + 1$, como se observa a continuación:

$$\frac{x^{15} + 1}{x^4 + x + 1} = x^{11} - x^8 - x^7 + x^5 + x^3 - x^2 - x - 1\,; \qquad res = 0.$$

Con lo anterior, se define que un polinomio irreducible $p(x)$, descrito por la ecuación 1.9.

$$p(x) = p_m x^m + p_{m-1} x^{m-1} + \cdots + p_1 x^1 + p_0 x^0, \tag{1.9}$$

de grado m con coeficientes $p_0, p_1, \ldots, p_{m-1}, p_m$, proveniente de $GF(2)$ es llamado polinomio primitivo si el entero positivo más pequeño n por el cual $p(x)$ divide exactamente a $x^n + 1$ es de valor igual a $2^m - 1$.

Es de resaltar que un polinomio primitivo sobre $GF(2)$ es siempre irreducible, mientras que un polinomio irreducible no siempre puede ser primitivo. Por ejemplo, el polimonio:

$$p(x) = x^4 + x^3 + x^2 + x + 1,$$

divide exactamente a $x^{15} + 1$, que cumple la condición $15 = 2^4 - 1$, pero este $p(x)$ también divide exactamente a $x^5 + 1$, donde $5 \neq 2^4 - 1$, por lo tanto $p(x) = x^4 + x^3 + x^2 + x + 1$ no es un polinomio primitivo sobre $GF(2)$.

1.2.4. Extensión del campo

La contrucción de una extensión de campo $GF(2^m)$ con $m > 1$, a partir del campo $GF(p)$, de un campo binario $GF(2)$, se realiza introduciendo un nuevo símbolo α y definiendo una multiplicación para introducir una secuencia de potencias de α.

Con la definición de multiplicación se tiene que:

$$0 \cdot \alpha^j = \alpha^j \cdot 0 = 0,$$
$$1 \cdot \alpha^j = \alpha^j \cdot 1 = \alpha^j,$$
$$\alpha^i \cdot \alpha^j = \alpha^j \cdot \alpha^i = \alpha^{i+j}.$$

El conjunto de elementos en los que se define la operación multiplicación está representado por el conjunto:

$$GF(2^m) = \left\{0, 1, \alpha, \alpha^2, \ldots, \alpha^{2^m - 2}\right\}.$$

Si el elemento α es una raíz del polimonio primitivo $p(x)$ de grado m sobre $GF(2)$, entonces $p(\alpha) = 0$ y teniendo en cuenta que $p(x)$ divide exactamente a $x^{2^m - 1} + 1$, se tiene:

$$x^{2^m - 1} + 1 = q(x)p(x),$$

evaluando en $x = \alpha$ se obtiene:

$$\alpha^{2^m-1} + 1 = q(\alpha)p(\alpha),$$

como $p(\alpha) = 0$, se tiene que:

$$\alpha^{2^m-1} + 1 = q(\alpha) \cdot 0,$$
$$\alpha^{2^m-1} + 1 = 0.$$

A continuación, se suma 1 en ambos lados, bajo suma módulo-2, obteniendo la ecuación 1.10.

$$\alpha^{2^m-1} = 1, \tag{1.10}$$

la condición anterior señala el carácter cíclico y finito del campo, debido a que la extensión del campo solo tiene los elementos indicados en la siguiente expresión:

$$GF(2^m) = \{0, 1, \alpha, \alpha^2, \dots, \alpha^{2^m-2}\}.$$

Una vez establecido que $p(x)$ es primitivo, se define α raíz del polinomio $p(x)$ tal que $p(\alpha) = 0$. Ejemplificándo lo anterior, con el polinomio $p(x) = x^4 + x + 1$, se tiene el siguiente resultado:

$$p(\alpha) = \alpha^4 + \alpha + 1,$$
$$0 = \alpha^4 + \alpha + 1,$$

sumando α^4 en ambos lados de la igualdad, se obtiene que:

$$\alpha^4 = \alpha + 1,$$

esta última expresión, se utiliza repetidamente para formar las representaciones de los polinomios para $GF(2^4)$, por ejemplo para α^5, α^6, α^7.

$$\alpha^5 = \alpha \cdot \alpha^4 = \alpha(\alpha + 1) = \alpha^2 + \alpha,$$
$$\alpha^6 = \alpha \cdot \alpha^5 = \alpha(\alpha^2 + \alpha) = \alpha^3 + \alpha^2,$$
$$\alpha^7 = \alpha \cdot \alpha^6 = \alpha(\alpha^3 + \alpha^2) = \alpha^4 + \alpha^3 = (\alpha + 1) + \alpha^3 = \alpha^3 + \alpha + 1.$$

De esta manera se obtienen α^8, α^9, α^{10}, α^{11}, α^{12}, α^{13}, α^{14}. En la tabla 1.2, se muestra la representación polinomial de cada potencia y su correspondiente formato binario, que más adelante se utiliza para la codificación BCH binaria.

Tabla 1.2 Representación polinomial de las raíces del campo $GF(2^4)$ dados por el polinomio primitivo $p(x) = x^4 + x + 1$.

Potencias	Polinomios	Binaria
$\mathbf{0}$	0	0000
$\mathbf{1}$	1	1000
$\boldsymbol{\alpha}$	α	0100
$\boldsymbol{\alpha^2}$	α^2	0010
$\boldsymbol{\alpha^3}$	α^3	0001
$\boldsymbol{\alpha^4}$	$1+\alpha$	1100
$\boldsymbol{\alpha^5}$	$\alpha+\alpha^2$	0110
$\boldsymbol{\alpha^6}$	$\alpha^2+\alpha^3$	0011
$\boldsymbol{\alpha^7}$	$1+\alpha+\alpha^3$	1101
$\boldsymbol{\alpha^8}$	$1+\alpha^2$	1010
$\boldsymbol{\alpha^9}$	$\alpha+\alpha^3$	0101
$\boldsymbol{\alpha^{10}}$	$1+\alpha+\alpha^2$	1110
$\boldsymbol{\alpha^{11}}$	$\alpha+\alpha^2+\alpha^3$	0111
$\boldsymbol{\alpha^{12}}$	$1+\alpha+\alpha^2+\alpha^3$	1111
$\boldsymbol{\alpha^{13}}$	$1+\alpha^2+\alpha^3$	1011
$\boldsymbol{\alpha^{14}}$	$1+\alpha^3$	1001

En las tablas A.2 y A.3 del anexo A, se indican las tablas de las representaciones polinómicas de las raíces del campo $GF(2^3)$ y $GF(2^5)$.

- **Polinomios mínimos**

En la extensión $GF(2^m)$ un polinomio irreducible $f(x)$ proveniente del campo $GF(2)$ posee al menos una raíz β, siendo β un elemento de esa extensión. Para hallar sus raíces en la extensión $GF(2^m)$ se hace uso del conjugado de β, β^{2^l}, con $l \geq 0$. De esta forma, todos los conjugados diferentes de β son raíces de $f(x)$. Si $f(\beta) = 0$ y $f\left(\beta^{2^l}\right) = 0$ entonces β^{2^l} es una raíz de $f(x)$ [3].

Se puede ejemplificar lo anterior con el polinomio $f(x) = x^6 + x^5 + x^4 + x^3 + 1$ del campo $GF(2)$, el cual tiene a α^4 como una de sus raíces en la extensión $GF(2^4)$.

$$f(x = \alpha^4) = (\alpha^4)^6 + (\alpha^4)^5 + (\alpha^4)^4 + (\alpha^4)^3 + 1,$$
$$f(x = \alpha^4) = \alpha^{24} + \alpha^{20} + \alpha^{16} + \alpha^{12} + 1,$$
$$f(x = \alpha^4) = 0.$$

Para verificar todas las raíces de este polinomio se aplica el conjugado β^{2^l} a α^4 de la siguiente manera:

$$(\alpha^4)^{2^0} = \alpha^4,$$
$$(\alpha^4)^{2^1} = \alpha^8,$$
$$(\alpha^4)^{2^2} = \alpha^{16} = \alpha,$$
$$(\alpha^4)^{2^3} = \alpha^{32} = \alpha^2,$$
$$(\alpha^4)^{2^4} = \alpha^{64} = \alpha^4.$$

De tal forma, que para valores $l > 3$, se vuelven a repetir las raíces.

En conclusión, los elementos de $GF(2^m)$ forman todas las raíces del polinomio $x^{2^m-1} + x$. Si cualquier elemento β, es raíz de $x^{2^m-1} + x$, β puede ser una raíz de cualquier polinomio sobre $GF(2)$ de grado menor a 2^m, entonces existe un $\varphi(x)$ de menor grado tal que $\varphi(\beta) = 0$, denominado *polinomio mínimo* de β, donde $\varphi(x)$ es irreducible.

Si $f(x)$ es irreducible sobre $GF(2)$, siendo β un elemento de $GF(2^m)$, asumiendo que $\varphi(x)$ es un polinomio mínimo de β y si $f(\beta) = 0$, se tiene la ecuación 1.11.

$$\varphi(x) = f(x). \tag{1.11}$$

Ahora se considera e como el entero positivo más pequeño tal que $\beta^{2^e} = \beta$, con $e \leq m$, como se indica en las ecuaciones 1.12 y 1.13.

$$f(x) = \prod_{i=0}^{e-l} (x + \beta^{2^i}), \tag{1.12}$$
$$\varphi(x) = \prod_{i=0}^{e-l} (x + \beta^{2^i}). \tag{1.13}$$

Con la ecuación 1.13 se puede generar todos los polinomios mínimos de un elemento de β proveniente del campo $GF(2^m)$.

En el siguiente ejemplo, se puede apreciar que para el elemento $\beta = \alpha^7$ de la extensión $GF(2^4)$, se puede generar el polinomio mínimo $\varphi(x)$.

$$(\alpha^7)^{2^0} = \alpha^7,$$
$$(\alpha^7)^{2^1} = \alpha^{14},$$
$$(\alpha^7)^{2^2} = \alpha^{28-15} = \alpha^{13},$$
$$(\alpha^7)^{2^3} = \alpha^{56-15(3)} = \alpha^{56-45} = \alpha^{11},$$
$$(\alpha^7)^{2^4} = \alpha^{112-15(7)} = \alpha^{112-105} = \alpha^7,$$

a partir de $l = 3$, se repite el elemento. Ahora el polinomio mínimo es:

$$\varphi(x) = (x + \alpha^7)(x + \alpha^{14})(x + \alpha^{13})(x + \alpha^{11}),$$
$$\varphi(x) = x^4 + x^3 + 1.$$

En el anexo B, se observa los polinomios mínimos de las extensiones $GF(2^3)$, $GF(2^4)$ y $GF(2^5)$.

1.3. CÓDIGOS BCH BINARIO

Los códigos bloque (n, k) a medida que se incrementa su longitud n, presentan los siguientes problemas:

- **Tiempo**: Cuando se incrementa la longitud del bloque, la duración en la formación de dicho bloque por los k bits también aumenta provocando un retardo de transmisión. Este efecto para aplicaciones reales como transmisión de voz no es tolerable.

- **Decodificación**: La complejidad aumenta con la longitud de los bloques, debido a que se debe buscar las palabras de código válidas para encontrar la que mejor se adapte con los bloques codificados.

Para contrarrestar los problemas mencionados, surgieron los llamados códigos BCH capaces de detectar y corregir múltiples errores para una longitud de palabra larga [4]. Estos se dividen en binarios y no binarios, dentro de los primeros se encuentran los códigos Hamming que tienen la capacidad de detectar hasta dos errores o corregir uno y en los segundos están los códigos Reed Solomon (RS), que se caracterizan por operar sobre múltiples bits [8].

Los códigos BCH binarios fueron descubiertos por Hocquenghem e independientemente por Bose y Chaudhuri entre 1959 y 1960, son unas de las clases más importantes de los codificadores de bloque lineales[7], los cuales cuentan con un esquema de decodificación eficiente. Además, son una subclase de los códigos cíclicos[8] de tipo binario, que permiten hacer un desplazamiento en lazo cerrado de una palabra código dando como resultado otra de éstas existente dentro del conjunto empleado para codificar los posibles mensajes [9]. De igual forma son códigos de corrección de t errores en el sentido de que pueden detectarlos y corregirlos de forma aleatoria por palabra código. Es importante mencionar que una propiedad importante

[7] Lineales: la suma módulo -2 de dos palabras de código es también otra palabra código.
[8] Códigos Cíclicos: son una subclase de los códigos bloques lineales.

es que son códigos sistemáticos, lo que les permite tener una ubicación de los bits de información en relación con los de redundancia.

Estos códigos reciben un paquete de información de longitud k, el cual se procesa convirtiéndolo en un bloque de longitud n, donde $n > k$, como se ve en la figura 1.2. De este modo, hay 2^k palabras código de tamaño n.

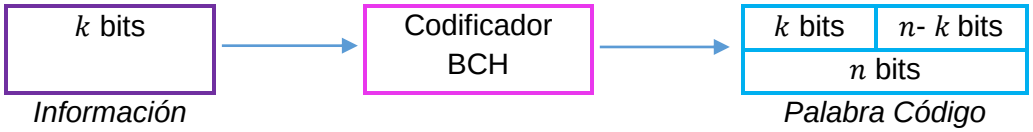


Figura 1.2 Representación de la formación de la palabra código.

Para cualquier entero positivo $m \geq 3$, existe un código BCH binario con los parámetros dados por la tabla 1.3.

Tabla 1.3 Parámetros de la codificación BCH binaria.

Longitud de bloque	$\boldsymbol{n = 2^m - 1}$
Número de dígitos de paridad	$n - k \leq mt$
Distancia mínima	$d_{min} \geq 2t + 1$

Este código puede corregir cualquier combinación de t errores o menos en un bloque $n = 2^m - 1$ dígitos binarios. A este código se le llama código BCH binario corrector de t errores y el polinomio generador de este código se especifica en términos de sus raíces provenientes del campo de Galois $GF(2^m)$.

1.3.1. Codificación BCH binario

La ecuación 1.14 determina el cálculo de la palabra código para un codificador sistemático, el cual se caracteriza por contener tal cual la palabra dato $d(x)$ en la propia palabra código $c(x)$, facilitando el proceso de la decodificación en el receptor, debido a que solo hay que ubicar el segmento donde inicia la palabra dato dentro de la palabra recibida $r(x)$ previamente ya corregida.

$$c(x) = x^{n-k} d(x) + b(x), \tag{1.14}$$

donde $b(x)$ está denotada por la ecuación 1.15.

$$b(x) = Res\left(\frac{x^{n-k} d(x)}{g(x)}\right), \tag{1.15}$$

el término x^{n-k} de las ecuaciones tiene como finalidad dar corrimiento a los bits de información y concatenar el bloque de redundancia. Se puede observar que la ecuación 1.14, cuenta con dos sumandos, el primero contiene el mensaje, en otras palabras, es el bloque de la información y el otro corresponde a los bits añadidos o bloque de redundancia $b(x)$ que contiene al polinomio generador [3].

- **Polinomio generador**

Siendo α un polinomio primitivo de $GF(2^m)$. El polinomio generador del código BCH binario, es el del grado más bajo sobre $GF(2)$ el cual tiene $\alpha, \alpha^2, \alpha^3, \dots, \alpha^{2t}$, como sus raíces, con $g(\alpha^i) = 0$ para $1 \leq i \leq 2t$.

Para obtener la palabra código es necesario conocer a que hace referencia el polinomio $g(x)$, el cual está dada por:

$$g(x) = m.c.m\{M_1(x), M_2(x), \dots, M_{2t}(x)\},$$

donde $M_i(x)$ es el polinomio mínimo de α^i.

Los códigos BCH se pueden denotar como $\mathrm{BCH}(n,k)$, es decir, un código cíclico $(n, n - grado[g(x)])$.

Con lo definido anteriormente, el proceso de codificación de un código BCH sobre $GF(2^m)$ de longitud n, capaz de corregir al menos t errores, consta de los siguientes pasos:

a) Determinar el polinomio primitivo sobre $GF(2^m)$, el cual da origen a la raíz primitiva de unidad β. Donde $n = 2^m - 1$.

b) Listar las potencias consecutivas de β: β^b, $\beta^{b+1}, \dots, \beta^{b+2t-1}$ con $b = 1$.

c) Determinar el polinomio mínimo de cada una de las potencias de β .

d) Hallar el $m.c.m$ de los polinomios mínimos, quien será en adelante el polinomio generador $g(x)$.

e) Formar la palabra código $c(x)$.

A continuación se da un ejemplo, del anterior proceso.

Inicialmente se toma el campo $GF(2^3)$, $m = 3$, una capacidad correctora de $t = 1$, y un dato $d(x) = 1 + x^2$, $d = [1010]^9$ entonces:

$$n = (2^3 - 1),$$
$$n = (2^3 - 1) = 7.$$

a) Siendo β un elemento primitivo y raíz del polinomio $x^3 + x + 1$ en $GF(2^3)$, cumple con $\beta^n = \beta^7 = 1$.

b) Las $2t$ potencias consecutivas de β se obtienen a partir de $t = 1$, donde $2t = 2$ potencias, que equivalen a $\{\beta, \beta^2\}$.

c) Los polinomios mínimos de $\{\beta, \beta^2\}$, para $1 + x + x^3$, corresponden a los dados por la tabla B.1 del Anexo B.

Para β, β^2; $M_1(x) = 1 + x + x^3$.

d) El cómputo del $m.c.m$ de estos polinomios mínimos da como resultado $g(x)$:

$$g(x) = M_1(x)$$
$$g(x) = 1 + x + x^3.$$

Entonces, $k = \big(n - grado\ del\ polinomio\ g(x)\big) = 7 - 3 = 4$; por lo tanto es un codificador $\mathrm{BCH}(7,4)$. Por medio de las ecuaciones. 1.14 y 1.15, se obtiene:

$$c(x) = x^{n-k}d(x) + b(x) = x^3 d(x) + b(x)\,,$$
$$d(x) = 1 + x^2,$$
$$b(x) = Res\left[\frac{x^3 \cdot (1 + x^2)}{1 + x + x^3}\right] = x^2,$$
$$c(x) = x^3(1 + x^2) + x^2 = x^2 + x^3 + x^5.$$

e) Finalmente en formato binario la palabra código es:

$$c = 001\mathbf{1010},$$

donde se observa que los cuatro dígitos menos significativos corresponden a los datos $d(x)$.

[9] Para las representaciones binarias, se tiene en cuenta que la potencia máxima del polinomio que representa al dato corresponde al bit menos significativo.

1.3.2. Decodificación BCH binario

El proceso de decodificación se puede llevar a cabo bajo el procedimiento mostrado en la Figura 1.3 [3].

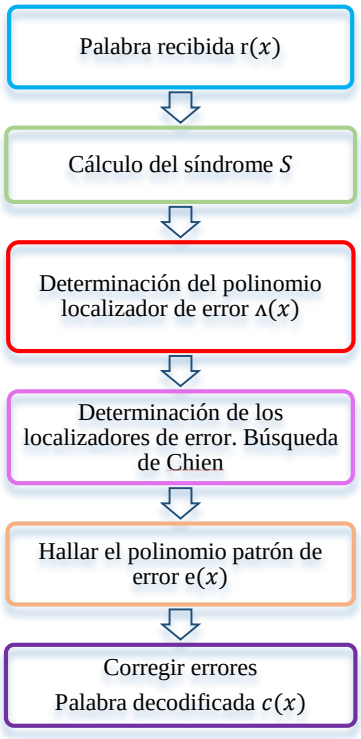


Figura 1.3 Proceso de decodificación de los códigos BCH binarios

Cada fase del proceso se explica a continuación:

a. Cálculo del Síndrome

Sean: la palabra de código $\mathbf{c} = [c_0, c_1, c_2, \ldots, c_{n-1}]$ enviada al canal con ruido y $\mathbf{r} = [r_0, r_1, r_2, \ldots, r_{n-1}]$, el vector palabra código recibido que contiene uno o varios errores. El vector suma $\mathbf{e} = \mathbf{r} + \mathbf{c}$ con $\mathbf{e} = [e_0, e_1, e_2, \ldots, e_{n-1}]$ es llamado el vector de error o patrón de error. Si $e_i = 1$ hay errores de transmisión en la posición i, causados por los efectos del canal en la palabra código, entonces $r_i \neq c_i$ y si $e_i = 0$ no hay errores en la palabra código recibido, entonces $r_i = c_i$. Se puede ver que el vector recibido r está dado por la ecuación 1.16.

$$\boldsymbol{r} = \boldsymbol{c} + \boldsymbol{e}. \tag{1.16}$$

Para verificar que el polinomio recibido contiene errores se debe evaluar el polinomio $r(x)$ para todos los valores de α^i, donde $1 \leq i \leq 2t$, con lo cual el resultado es la expresión conocida como *Síndrome,* definida por la ecuación 1.17.

$$s_i = r(\alpha^i). \tag{1.17}$$

el valor de α^i, corresponde a todas las raíces provenientes del campo $GF(2^m)$. Si $r(\alpha^i) = 0$, se afirma que el mensaje está libre de errores y por tanto $c(\alpha^i) = 0$. De esta manera,

$$r(\alpha^i) = c(\alpha^i) + e(\alpha^i),$$
$$r(\alpha^i) = 0 + e(\alpha^i) = s_i,$$

Así para $1 \leq i \leq 2t$, se obtiene el vector de síndromes:

$$\boldsymbol{S} = [s_1, s_{2,}, \dots\ s_{2t}]. \tag{1.18}$$

De la ecuación 1.18 se puede deducir que el síndrome depende del patrón de error que se introduce en la palabra código. Por otro lado, con un número determinado de errores v en la palabra código y suponiendo que las posiciones de los errores es $\{x^{j_1}, x^{j_2}, \dots, x^{j_v}\}$ se tiene la ecuación 1.19.

$$e(x) = x^{j_1} + x^{j_2} + \dots + x^{j_v}\,. \tag{1.19}$$

De las ecuaciones 1.18 y 1.19 se obtiene el siguiente sistema de ecuaciones 1.20.

$$\begin{aligned} s_1 &= \alpha^{j_1} + \alpha^{j_2} + \dots + \alpha^{j_v}, \\ s_2 &= (\alpha^{j_1})^2 + (\alpha^{j_2})^2 + \dots + (\alpha^{j_v})^2, \\ s_3 &= \left(\alpha^{j_1}\right)^3 + \left(\alpha^{j_2}\right)^3 + \dots + \left(\alpha^{j_v}\right)^3, \\ &\vdots \\ s_{2t} &= (\alpha^{j_1})^{2t} + (\alpha^{j_2})^{2t} + \dots + (\alpha^{j_v})^{2t}\,, \end{aligned} \tag{1.20}$$

El método que soluciona este sistema de ecuaciones es considerado un algoritmo de decodificación para los códigos BCH binario. En general, el sistema de ecuaciones 1.20, se define como:

$$s_i = \sum_{l=1}^{v} \left(\alpha^{j_l}\right)^i \qquad i = 1,2,3, \dots, 2t. \tag{1.21}$$

Una vez, $\alpha^{j_1} + \alpha^{j_2} + \dots + \alpha^{j_v}$ se ha encontrado, las potencias $j_1, j_2, \dots, j_v$, especifican las ubicaciones de error en $e(x)$. La ecuación 1.21 tiene muchas soluciones posibles, por lo cual de cada solución se obtiene un patrón de error diferente.

b. Encontrar el polinomio localizador de error.

Al hacer $\beta_l = \alpha^{j_l}$, para $1 \leq l \leq v$ y sustituyendo en el sistema de ecuaciones 1.20, se obtiene la ecuación 1.22.

$$\begin{aligned} s_1 &= \beta_1 + \beta_2 + \ldots + \beta_v, \\ s_2 &= \beta_1^2 + \beta_2^2 + \ldots + \beta_v^2, \\ s_3 &= \beta_1^3 + \beta_2^3 + \ldots + \beta_v^3, \\ &\vdots \\ s_{2t} &= \beta_1^{2t} + \beta_2^{2t} + \ldots + \beta_v^{2t}. \end{aligned} \tag{1.22}$$

Los β_l recibe el nombre de *números localizadores de error*. Las $2t$ ecuaciones son funciones simétricas en $\beta_1, \beta_2, \ldots \beta_v$, conocidas como funciones simétricas de sumas de potencia [2]. De esta forma se define el polinomio especificado en la ecuación 1.23.

$$\begin{aligned} \Lambda(x) &= \prod_{l=1}^{v} (1 - \beta_l x) \\ \Lambda(x) &= [(1 + \beta_1 x)(1 + \beta_2 x) \ldots (1 + \beta_v x)] \\ \Lambda(x) &= \Lambda_0 + \Lambda_1 x + \Lambda_2 x^2 + \ldots + \Lambda_v x^v \end{aligned} \tag{1.23}$$

Las raíces de $\Lambda(x)$: ${\beta_1}^{-1}$, ${\beta_2}^{-1}$, ${\beta_v}^{-1}$ son inversos de los números localizadores de error, por esto a $\Lambda(x)$ se le conoce como el *polinomio localizador de error*, donde los coeficientes son los que se deben encontrar para realizar la corrección de errores. Los números localizadores de error y los coeficientes de $\Lambda(x)$ se relacionan mediante la ecuación 1.24.

$$\begin{aligned} \Lambda_0 &= 1, \\ \Lambda_1 &= \beta_1 + \beta_2 + \beta_3 + \ldots + \beta_v, \\ \Lambda_2 &= \beta_1\beta_2 + \beta_2\beta_3 + \ldots + \beta_{v-1}\beta_v, \\ &\vdots \\ \Lambda_v &= \beta_1\beta_2\beta_3 \ldots \beta_v, \end{aligned} \tag{1.24}$$

teniendo en cuenta que $\Lambda_0 = 1$ se obtiene la ecuación 1.25.

$$\Lambda(x) = 1 + \Lambda_1 x + \Lambda_2 x^2 + \ldots + \Lambda_v x^v. \tag{1.25}$$

Con las ecuaciones 1.22 y 1.24 se puede deducir la relación existente entre el síndrome s_i y los coeficientes de $\Lambda(x)$, de la siguiente manera:

$$\begin{aligned} s_1 + \Lambda_1 &= 0, \\ s_2 + s_1\Lambda_1 + 2\Lambda_2 &= 0, \\ s_3 + s_2\Lambda_1 + s_1\Lambda_2 + 3\Lambda_3 &= 0, \\ s_v + s_{v-1}\Lambda_1 + \cdots + s_1\Lambda_{v-1} + v\Lambda_v &= 0, \\ &\vdots \\ s_{v+1} + s_v\Lambda_1 + \cdots + s_2\Lambda_{v-1} + s_1\Lambda_v &= 0. \end{aligned} \tag{1.26}$$

Este grupo de ecuaciones 1.26 se describen como Identidades de Newton[10]. A partir de éstas se encuentran los coeficientes de $\Lambda(x)$ y al encontrar sus raíces se determinan los números localizadores de error. Es importante, precisar que se debe encontrar el $\Lambda(x)$ de menor grado, debido a que éste, produce un patrón de error con el menor número de errores (evento de mayor probabilidad) [10].

Existen múltiples algoritmos que se pueden utilizar para hallar el polinomio localizador de error entre los que se destacan los siguientes:

- Algoritmo de Berlekamp-Massey
- Algoritmo Euclidiano.
- Algoritmo de Peterson.
- Técnica basada en transformadas de Fourier para campos $GF(p)$.

Para el presente trabajo, el algoritmo que se utiliza es el Berlekamp-Massey, que a diferencia de los otros, tiene una menor complejidad matemática y está diseñado para optimizar recursos en el momento de hacer una implementación hardware.

Algoritmo de Berlekamp-Massey

Este algoritmo también mencionado como ABM, permite que el proceso de implementación de la etapa de decodificación de los códigos BCH sea eficiente, dado que la resolución del sistema de ecuaciones 1.26 a nivel computacional es posible [11] [12].

El sistema de ecuaciones 1.26 se puede visualizar matricialmente de la siguiente forma:

$$\begin{bmatrix} s_1 & s_2 & s_3 & \cdots & s_v \\ s_2 & s_3 & s_4 & \cdots & s_{v+1} \\ s_3 & s_4 & s_5 & \cdots & s_{v+2} \\ \vdots & \vdots & & & \\ s_v & s_{v+1} & s_{v+2} & & s_{2v-1} \end{bmatrix} \begin{bmatrix} \Lambda_v \\ \Lambda_{v-1} \\ \Lambda_{v-2} \\ \vdots \\ \Lambda_1 \end{bmatrix} = - \begin{bmatrix} s_{v+1} \\ s_{v+2} \\ s_{v+3} \\ \vdots \\ s_{2v} \end{bmatrix}, \tag{1.27}$$

[10] Identidades de Newton: Relaciona las sumas de potencia con los polinomios simétricos elementales.

Para los códigos BCH binarios la solución de la ecuación 1.26, se desarrolla partiendo del sistema de ecuaciones 1.27. De tal forma se cumple que:

$$S_j = -\sum_{l=1}^{v} \Lambda_l S_{j-l}, \tag{1.28}$$

para $j = v+1, v+2, \dots, 2v$.

De la ecuación 1.28, se puede realizar su implementación utilizando los *R*egistros de Desplazamiento con Retroalimentación Lineal (LFSR, *Linear Feedback Shift Register)*. Manteniendo Λ fijo, estos circuitos generan una secuencia con los síndromes encontrados. Como se desea encontrar el polinomio $\Lambda(x)$, de menor grado posible, se debe encontrar el LFSR de menor peso en todos los registros de corrimiento que se generan.

A continuación, se establece una relación de recurrencia entre los diferentes parámetros de la ecuación 1.28 y se calculan dos valores, uno correspondiente al polinomio de conexión que en este caso es el polinomio $\Lambda(x)$, y el otro es respectivamente la longitud L, es decir, un polinomio dado por la ecuación 1.25, donde los coeficientes son los que aparecen en la recurrencia dada, y donde L está dada por el grado del polinomio $\deg(\Lambda(x)) \leq L$.

El algoritmo BM es un proceso inductivo donde en el μ-ésimo paso se crean dos elementos, L_μ y ${\Lambda^\mu}_{(x)}$ donde el segundo término corresponde al polinomio cuyos coeficientes generan los primeros μ síndromes. De esta forma se busca que en la iteración μ, a partir de la $\mu-1$-ésima del polinomio, se calcule la salida dada por la ecuación 1.29.

$$\hat{S}_\mu = \sum_{l=1}^{L_{\mu-1}} {\Lambda_l}^{\mu-1} S_{\mu-l}. \tag{1.29}$$

Al restar la ecuación 1.29 a la salida esperada S_μ, $\left(S_\mu - \hat{S}_\mu\right)$ se obtiene la ecuación 1.30.

$$d_\mu = \sum_{l=0}^{L_{\mu-1}} {\Lambda_l}^{\mu-1} S_{\mu-l}. \tag{1.30}$$

La expresión d_μ, se llama la $\mu-1$-ésima discrepancia. Si el polinomio no se modifica significa que $d_\mu = 0$. Si ocurre lo contrario, $d_\mu \neq 0$, el polinomio queda expresado por la ecuación 1.31.

$$\Lambda^\mu(x) = \Lambda^{(\mu-1)}(x) - {d^{-1}}_m d_\mu x^{\mu-m} \Lambda^{(m-1)}(x), \tag{1.31}$$

donde m, representa la última iteración por la cual se modificó el polinomio de conexión.

A continuación, en la tabla 1.4 se describen las diferentes iteraciones del algoritmo BM.

Sean,

- L_μ: Grado del polinomio localizador de error $\Lambda^{\mu}(x)$ para el paso por la $\mu-1$-ésima iteración.
- d_μ: $\mu-1$-ésima discrepancia.
- $T_{(x)}$: Nuevo polinomio de conexión para el caso $d_\mu = 0$.
- $B_{(x)}$: Antiguo polinomio de conexión.
- j: Ubicación del símbolo más antiguo dentro del LFSR a partir de la posición donde éste falló en su secuencia, teniendo en cuenta que los coeficientes de los polinomios de conexión determinan las derivaciones de los registros de corrimiento.

Tabla 1.4 Iteraciones del algoritmo Berlekam-Massey.

Inicialización $\boldsymbol{\mu = 0}$	El producto de $\boldsymbol{x\Lambda^0(x)}$ se almacena en la variables $\boldsymbol{B_{(x)}}$ temporalmente**.**	$\boldsymbol{L_{\mu=0} = 0}$, $\boldsymbol{B_{(x)} = x}$, $\boldsymbol{\Lambda^{\mu=0}(x) = 1}$, $\boldsymbol{j = 0}$
Primera Iteración $\boldsymbol{\mu = 1}$	Se calcula d_μ, a partir de los coeficientes de las derivaciones y del contenido de los LFSR, cuando el síndrome es procesado.	$d_\mu = S_\mu + \sum_{l=1}^{L_{\mu-1}} \Lambda_l S_{\mu-l}$
Segunda Iteración $\boldsymbol{\mu = 2}$ $\boldsymbol{L_{\mu=1}}$ $\boldsymbol{j = 1}$	Se mira si hay discrepancia. • Si $d_\mu = 0$, los coeficientes de las derivaciones generan la secuencia del síndrome dado. $B_{(x)}$ es actualizado por la siguiente iteración y los valores de j y L_μ permanecen constantes.	$B_{(x)} \leftarrow xB_{(x)}$
	• Si $d_\mu \neq 0$, se realiza el cómputo del nuevo polinomio de conexión $T_{(x)}$ debido a que no se genera la secuencia del síndrome dado. Cuando se calcula $T_{(x)}$ se observa si debe ser extendido el registro de corrimiento: ▪ Si $L_\mu < \mu - j$, $B(x)$ es normalizado y actualizado y las derivaciones de los LFSR son modificadas por los coeficientes de $T_{(x)}$. También se modifican la longitud de $T_{(x)}$, j y L_μ. ▪ Si $L_\mu \geq \mu - j$ las derivaciones del LFSR es modificado por los coeficientes de $T_{(x)}$ y $B(x)$ es actualizado para la siguiente iteración.	$T_{(x)} = \Lambda^{\mu-l}(x) + d_\mu B(x)$
⋮	Se siguen realizando las iteraciones si y solo si $\mu \neq 2t$.	
Última Iteración $\boldsymbol{\mu = 2t}$	El algoritmo se detiene al finalizar la iteración $\mu = 2t$, entonces es calculado el *polinomio localizador de error*.	$\Lambda^{\mu=2t}(x)$= *Polinomio localizador de error.*

c. Hallar las posiciones de los errores

Una vez determinado el polinomio $\Lambda(x)$ se calculan sus raíces utilizando la *búsqueda de Chien* que facilita la factorización de los polinomios sobre GF.

Retomando la ecuación 1.25, para v errores, se evalúa el polinomio localizador de error en cada elemento no nulo del campo de Galois, esto es:

$$\{1, \alpha, \alpha^2, \cdots, \alpha^{2^m-2}\}. \tag{1.32}$$

Se obtiene:

$$\begin{gathered}\Lambda(1) = 1 + \Lambda_1(1) + \Lambda_2(1)^2 + \ldots + \Lambda_v(1)^v, \\ \Lambda(\alpha) = 1 + \Lambda_1(\alpha) + \Lambda_2(\alpha)^2 + \ldots + \Lambda_v(\alpha)^v, \\ \vdots \\ \Lambda\left(\alpha^{2^m-2}\right) = 1 + \Lambda_1(\alpha^{2^m-2}) + \Lambda_2(\alpha^{2^m-2})^2 + \ldots + \Lambda_v\left(\alpha^{2^m-2}\right)^v.\end{gathered} \tag{1.33}$$

Si las raíces residen en el mismo campo y son diferentes se pueden utilizar para encontrar las posiciones de los errores.

d. Encontrar el polinomio patrón de error $e(x)$.

A partir de las raíces β_l, $1 \leq l \leq v$ se obtienen los recíprocos $\beta_l{}^{-1}$ para formar el polinomio patrón de error $e(x)$, según la ecuación 1.19.

e. Corregir los errores en $r(x)$.

Una vez conocido $e(x)$, se suma con $r(x)$ obteniendo la palabra de código transmitida $c(x)$, de la siguiente forma:

$$c(x) = r(x) + e(x). \tag{1.34}$$

Para ilustrar lo anterior, se retoma el ejemplo del proceso de la codificación descrito en la sección 1.3.1. En la tabla 1.5 se describen los parámetros.

Tabla 1.5 Parámetros para la Decodificación BCH binaria.

Codificador BCH binario	$(7, 4)$
Número de errores	: $t = 1$
Polinomio primitivo	: $p(x) = 1 + x + x^3$
Polinomio generador	: $g(x) = 1 + x + x^3$
Palabra código	: $c(x) = x^2 + x^3 + x^5$; $\boldsymbol{c} = 0011010$
Polinomio recibido	: $r(x) = x^2 + x^3 + x^5 + x^6$; $\boldsymbol{r} = 0011011$

- **Cálculo del síndrome**

Para encontrar los $2t = 2$ componentes del síndrome, se hace la evaluación de los α^i, $i = 1,2$, en el polinomio recibido $r(x) = x^2 + x^3 + x^5 + x^6$; de esta forma se obtiene de $s_i = r(\alpha^i)$, dos síndromes:

$$s_1 = r(\alpha) = \alpha^2 + \alpha^3 + \alpha^5 + \alpha^6 = \alpha^6,$$
$$s_2 = r(\alpha^2) = \alpha^4 + \alpha^6 + \alpha^{10} + \alpha^{12} = \alpha^5.$$

- **Determinación del polinomio localizador de error $\Lambda(x)$**

Con base en los distintos síndromes previamente calculados, se aplica el algoritmo BM como se muestra a continuación.

✓ **Iteración 1:**

a) Inicializador:

$\mu = 0$, $L_0 = 0$, $B_{(x)} = x$, $\Lambda^0(x) = 1$, $j = 0$
$\mu = \mu + 1$, $\mu = 1$.

b) Cómputo del error en el siguiente síndrome:

$$d_\mu = s_\mu + \sum_{l=1}^{L_{\mu-1}} \Lambda_l s_{\mu-l},$$
$$d_1 = s_1 + \sum_{l=1}^{0} \Lambda_1 s_{\mu-1} = s_1 + \Lambda_1 s_0,$$

De la ecuación 1.25 se obtienen los valores de Λ; para este caso $\Lambda_1 = 0$, entonces:

$$d_1 = s_1 + 0 \cdot s_0 = s_1 = \alpha^6,$$

como $d = \alpha^6 \neq 0$, significa que hay discrepancia, por la tanto se debe hacer el cómputo de $T(x)$.

$$T_{(x)} = \Lambda^{\mu-l}(x) + d_\mu B(x),$$

como $\mu = 1$, $B(x) = x$ y $d_1 = \alpha^6$.

$$T_{(x)} = \Lambda^0(x) + d_1 B(x),$$
$$T_{(x)} = 1 + \alpha^6 x.$$

Ahora se procede a verificar si:

$$\begin{aligned} L_{\mu-1} &< \mu - j, \\ L_0 &< 1 - 0, \\ 0 &< 1. \end{aligned}$$

Como existe discrepancia, se procede a normalizar y actualizar a $B(x)$ y las derivaciones de los LFSR son modificadas por los coeficientes de $T_{(x)}$.

c) Extensión del registro de corrimiento

Primero se almacena el antiguo registro de corrimiento y se normaliza $B_{(x)}$

$$\begin{aligned} B_{(x)} &\leftarrow (d_\mu)^{-1}\Lambda^{\mu-l}(x), \\ B_{(x)} &\leftarrow (d_1)^{-1}\Lambda^{0}(x), \\ B_{(x)} &= (\alpha^6)^{-1} \cdot 1 = \alpha. \end{aligned}$$

Ahora se actualiza los registros de corrimiento

$$\begin{aligned} \Lambda^{\mu}(x) &\leftarrow T(x), \\ \Lambda^{1}(x) &= 1 + \alpha^6 x. \end{aligned}$$

Por último se actualiza la longitud

$$\begin{aligned} T_\mu &\leftarrow \mu - j, \\ j &\leftarrow \mu - L_{\mu-1}, \\ L_\mu &\leftarrow T_\mu. \end{aligned}$$

Haciendo los reemplazos tenemos

$$\begin{aligned} T_1 &= 1 - 0 = 1, \\ j &= 1 - 0 = 1, \\ L_1 &= 1. \end{aligned}$$

d) Obtención del polinomio actualizado

$$\begin{aligned} B_{(x)} &\leftarrow xB_{(x)}, \\ B_{(x)} &= x\alpha. \end{aligned}$$

Ahora se debe verificar si $\mu = 2t$. Como $\mu = 1$ y $2t = 2$ la igualdad no se cumple, entonces se debe seguir con la siguiente iteración partiendo de los valores obtenidos en la iteración anterior.

✓ **Iteración 2**

a) Inicializadores con iteración anterior

$$\mu = 1;\ B_{(x)} = x\alpha,\ \Lambda^1(x) = 1 + \alpha^6 x,\ L_1 = 1,\ j = 1$$
$$\mu = \mu + 1,$$
$$\mu = 2,$$

b) Cómputo del error en el siguiente síndrome

$$d_\mu = s_\mu + \sum_{l=1}^{L_{\mu-1}} \Lambda_l s_{\mu-l},$$
$$d_2 = s_2 + \sum_{l=1}^{1} \Lambda_1 s_{\mu-1} = s_2 + \Lambda_1 s_1 = \alpha^5 + \alpha^6 * \alpha^6,$$
$$d_2 = \alpha^5 + \alpha^5 = 0,$$

como $d = 0$ significa que no hay discrepancia, por la tanto se procede a obtener el polinomio actualizado.

c) Obtención del polinomio actualizado

$$B_{(x)} \leftarrow xB_{(x)},$$
$$B_{(x)} = xB_{(x);}\ B_{(x)} = \alpha x,$$
$$B_{(x)} = \alpha x^2.$$

Ahora se debe verificar si $\mu = 2t$, Como $\mu = 2$ y $2t = 2$, entonces significa que las iteraciones del algoritmo BM finalizan.

A continuación la tabla 1.6 resume los datos entregados por las diferentes iteraciones, con el fin de visualizar el polinomio localizador de error $\Lambda(x)$.

Tabla 1.6 Resultados obtenidos del algoritmo Berlekamp Massey.

$\boldsymbol{\mu}$	$\boldsymbol{d_\mu}$	$\boldsymbol{B_{(x)}}$	$\boldsymbol{\Lambda^\mu(x)}$	$\boldsymbol{j}$	$\boldsymbol{L_\mu}$
0	α^6	x	1	0	0
1	α	αx^2	$1 + \alpha^6 x$	1	1

De este algoritmo, el polinomio localizador del error es $\Lambda(x) = 1 + \alpha^6 x$.

- **Determinación de los localizadores de error. Búsqueda de Chien**

Para determinar las raíces por medio de la búsqueda de Chien; se deben evaluar los elementos no nulos de $GF(2^3)$ que corresponden a $\{1, \alpha, \alpha^2, \alpha^3, \cdots, \alpha^7\}$ en el polinomio localizador y ubicar aquellos que den como resultado $\Lambda = 0$. Entonces:

$$\Lambda(1) = 1 + \alpha^6 = \alpha^2.$$
$$\Lambda(\alpha) = 1 + \alpha^7 = 0.$$
$$\Lambda(\alpha^2) = 1 + \alpha^8 = 1 + \alpha.$$
$$\Lambda(\alpha^3) = 1 + \alpha^9 = 1 + \alpha^2.$$
$$\Lambda(\alpha^4) = 1 + \alpha^{10} = \alpha.$$
$$\Lambda(\alpha^5) = 1 + \alpha^{11} = 1 + \alpha^2 + \alpha.$$
$$\Lambda(\alpha^6) = 1 + \alpha^{12} = \alpha^2.$$

La raíz encontrada corresponde a $\Lambda(\alpha)$**.**

- **Determinación polinomio patrón del error $e(x)$.**

Una vez se obtienen los valores de las raíces, se calculan sus recíprocos, debido a que sus inversos indican la posición de los errores dentro de la palabra recibida $r(x)$.

$$\Lambda(\alpha) = \frac{1}{\alpha} = \frac{\alpha^7}{\alpha} = \alpha^6.$$

Estos valores encontrados son las ubicaciones de los errores, por lo tanto el polinomio patrón de error es:

$$e(x) = x^6.$$

- **Corrección del error**

Se realiza la suma del polinomio patrón del error y el polinomio recibido:

$$c(x) = e(x) + r(x),$$
$$c(x) = x^6 + (x^2 + x^3 + x^5 + x^6),$$
$$c(x) = x^2 + x^3 + x^5.$$

Finalmente la palabra que se transmitió fue:

$$c(x) = x^2 + x^3 + x^5,$$
$$\boldsymbol{c} = 0011010.$$

En el Apéndice A, se muestra un ejemplo con mayor capacidad correctora del proceso de codificación y decodificación.

CAPÍTULO 2

MODELADO, SIMULACIÓN E IMPLEMENTACIÓN

2.1. METODOLOGÍA DE TRABAJO

Para cumplir con los objetivos planteados se toma de referencia el modelo Bob Zeiman [13], en el cual se destacan siete fases, expuestas en la figura 2.1.

Figura 2.1 Metodología de Trabajo.

Las anteriores fases están distribuidas en las secciones 2.2, 2.3 y 2.4 en las cuales se profundiza sobre el tema.

2.2. ANÁLISIS DE REQUERIMIENTOS

En esta sección se describen los requisitos funcionales del proyecto de grado, teniendo en cuenta los objetivos y la búsqueda de diferentes tecnologías hardware y software existentes en el mercado actual, las cuales permiten la implementación del sistema de comunicaciones.

2.2.1. Fase 0. Descripción de las especificaciones

2.2.1.1. Requisitos del proyecto

Los requisitos del proyecto "Análisis del desempeño de un sistema de comunicaciones con codificación BCH binaria basado en hardware reconfigurable" se plantean a continuación:

- Identificar y analizar las características técnicas asociadas a los esquemas de codificación BCH binaria.
- Implementar un sistema de comunicaciones basado en hardware reconfigurable con el esquema de codificación BCH binaria, con un canal de Ruido Gausiano Aditivo Blanco (AWGN, *Additive White Gaussian Noise*), y los esquemas de modulación FSK y MSK implementados en la fase 1.
- Evaluar el desempeño de un sistema de comunicaciones banda base con codificación BCH binaria con modelo de canal AWGN sobre hardware reconfigurable, teniendo en cuenta las curvas teórica y práctica de la BER y la Tasa de Error de Palabra (WER, *Word Error Rate*).

2.2.1.2. Búsqueda de tecnologías hardware y software

En el mercado actual se destacan diversas tecnologías software y hardware, las cuales se utilizan para múltiples aplicaciones. A continuación, se mencionan algunas de estas con sus respectivas características y capacidades:

- **Tecnología Hardware**

FPGA: Creado por Ross Freeman, el cofundador de Xilinx, a mediados de 1985. Estos dispositivos utilizan bloques de lógica pre-construidos y recursos para ruteo programables, se configuraran para implementar funcionalidades personalizadas. Es importante mencionar que para todos sus componentes, solo se deben desarrollar tareas de cómputo digital en software y compilarlas en un archivo de configuración o

bitstream que contiene información de las conexiones. Además, los FPGAs son completamente reconfigurables debido a que en el momento de compilar diferentes configuraciones de circuitos, adoptan nuevas funcionalidades.

La adopción de chips FPGAs ha sido impulsada porque éstos combinan lo mejor de los Circuitos Integrados de Aplicación Específica (ASIC, A*pplication-Specific Integrated Circuit*) y de los sistemas basados en procesadores. Así mismo ofrecen velocidades temporizadas por hardware y fiabilidad, sin requerir altos volúmenes de recursos para compensar el gran gasto que genera un diseño personalizado de ASIC. A diferencia de los procesadores, los FPGAs llevan a cabo diferentes operaciones de manera paralela, por tal razón no necesitan competir por los mismos recursos. Cada tarea de procesos independientes se asigna a una sección dedicada del chip, y puede ejecutarse de manera autónoma sin ser afectada por otros bloques; como resultado, el rendimiento de una parte de la aplicación no se ve afectada cuando se agregan otros procesos.

Las especificaciones del chip FPGA, generalmente están divididos en bloques de lógica configurables como segmentos o células de lógica, funciones fijas de lógica como multiplicadores y recursos de Memoria de Acceso Aleatorio (RAM, *Random Access Memory*) en bloques embebidas, que permiten seleccionar y comparar FPGAs para una aplicación en particular [14].

Los principales fabricantes y distribuidores de dispositivos lógicos programables son las empresas de Xilinx y Altera, quienes se han consolidado en el mercado por su desarrollo. Entre las familias de Xilinx están XC Virtex y XC Spartan. La primer serie contiene dispositivos de gama alta, que ofrece velocidades de trabajo hasta de 600 MHz, un alto número de compuertas y bajo consumo de potencia, sin embargo su costo es elevado, debido a que son adecuados para sistemas complejos que requieran un alto desempeño. Por otro lado la familia Spartan, es menos costosa que la anterior; esta proporciona un buen desempeño en los sistemas implementados y es muy utilizada en diseños que no requieran un número elevado de celdas lógicas [2].

USRP (Universal Software Radio Peripheral): Está diseñado para trabajar en conjunto con un computador a través de un FPGA para la realización de radio software. Este periférico realiza las funciones de llevar la señal a banda base de Radio Frecuencia (RF, *Radio Frecuency*), a través de la sección de Frecuencia Intermedia (IF, *Intermediate Frecuency)* y viceversa, para un sistema común de radio comunicaciones como se ve en la figura 2.2.

.

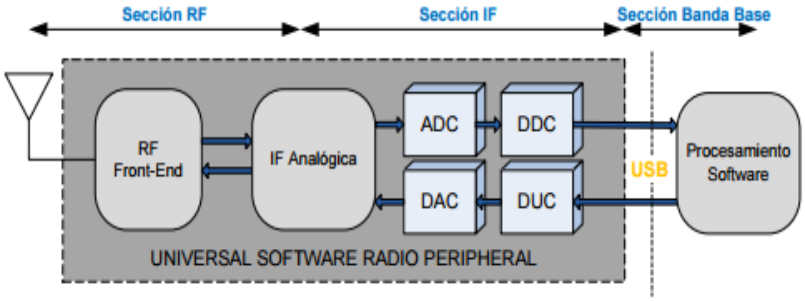


Figura 2.2 Bloque Principal en un sistema USRP. Fuente: [15].

El USRP cuenta con dos niveles de tarjetas: el primero, es la tarjeta base en la cual se encuentra el FPGA, los Convertidores Analógicos a Digital (ADC, *Analog to Digital Converter*) y los Convertidores Digital a Analógico (DAC, *Digital to Analog Converter*) conectados a un Convertidor Digital de Bajada (DDC, *Digital Down Converter*), al Convertidor Digital de subida (DUC, *Digital Up Converter*) y a la alimentación y conexión vía Bus Serial Universal (USB, *Universal Serial Bus)*; el segundo, se compone de tarjetas secundarias. Éstas existen para transmisión y recepción y trabajan como RF *front-end* las cuales llevan la señal hasta la banda de RF deseada [15].

En el mercado actual existen gran variedad de dispositivos USRP con los cuales se pueden lograr diferentes aplicaciones en el campo de la industria, en el comercio y las telecomunicaciones, entre otros. Los dispositivos para fines académicos se pueden clasificar dependiendo del entorno de desarrollo de la siguiente manera:

- USRP2.
- USRP Series Red (N210, N200).
- Serie USRP autobús (B200, B210).
- USRP X Series (X310, X300) que se apoyan en MATLAB y Simulink.

RASPBERRY PI: Computador de bajo costo, tamaño pequeño, que apareció cerca del año 2012. Existen dos modelos el A y el B, este último, cuenta con un chip integrado que contiene un procesador con varias frecuencias de funcionamiento, un procesador gráfico y distintas cantidades de memoria RAM [16].

Con el soporte que tiene Matlab para este dispositivo, se puede acceder a diferentes periféricos. Este apoyo permite adquirir datos de diferentes sensores y dispositivos de imagen conectados a la Raspberry Pi [17].

ARDUINO: Es una plataforma de prototipos electrónica de código abierto basada en hardware y software flexibles y fáciles de usar, el hardware es de tamaño pequeño con el que se puede leer información de diferentes dispositivos electrónicos [18].

Al igual que la Raspberry Pi, cuenta con un soporte de paquetes de Matlab para el hardware Arduino para comunicarse con él, a través de un cable USB. Este paquete se basa en un programa de servidor, que escucha los comandos que llegan a través del puerto serie, los ejecuta y si es necesario, devuelve un resultado [19].

- **Herramientas Software**

MATLAB: Herramienta de software matemático, con un lenguaje de alto nivel y un entorno interactivo que permite hacer cálculos numéricos, analizar-visualizar datos, programar-desarrollar algoritmos, crear interfaces de usuario y permitir la comunicación con programas de diferentes lenguajes y otros dispositivos hardware; además cuenta con una herramienta Simulink, para el diseño de sistemas de comunicaciones. Está disponible para plataformas Windows, Mac OS, Unix, GNU/Linux [20].

OCTAVE: Es un software libre bajo la licencia GNU[11], presenta un lenguaje de alto nivel que permite resolver problemas numéricos de algebra lineal, cálculo de raíces de ecuaciones lineales o no lineales, integración de ecuaciones diferenciales ordinarias y diferenciales algebraicas, cuenta con una interfaz sencilla, orientada a la línea de comandos y presenta compatibilidad con programas similares como Matlab y Scilab [21].

SCILAB: Es un paquete de software libre de código abierto para computación científica, orientada al cálculo numérico, a las operaciones matriciales y especialmente a las aplicaciones de ingeniería. Es similar a Matlab y otros programas de cálculo numérico. Puede ser utilizado en variedad de sistemas operativos como Unix, Windows, Linux [22].

DSP (Digital Signal Processing)-Builder: Los Procesadores Digitales de Señal Builder de Altera integran dos herramientas: la primera, de alto nivel para desarrollo de algoritmos y la segunda, el Lenguaje de Descripción Hardware (HDL, *Hardware Description Language)* a través de la combinación del desarrollo de algoritmos, simulación y verificación de las capacidades de las herramientas de diseño de Mathworks.

Consta de un conjunto de bloques de construcción que acorta los ciclos de diseño DSP, ayudando a crear la representación hardware en un entorno de desarrollo amigable. Permite utilizar las funciones de Matlab y los bloques de Simulink existentes como parte del proceso de diseño y prueba, permitiendo evaluar rápidamente el desempeño de los diseños en dispositivos de Altera, generar código HDL optimizado en tiempo y verificar las implementaciones de hardware con modelos de Simulink.

[11] GNU: Es un sistema operativo de software libre.

Con todo lo anterior, DSP Builder ofrece bloques en Simulink capaces de desarrollar tareas básicas como funciones aritméticas y de almacenamiento, así como funciones complejas de corrección de errores y filtrado [23].

System Generator Xilinxs (XGS): Es una herramienta de diseño de sistemas de alto nivel que permite el desarrollo y verificación de algoritmos DSP, optimizada para FPGAs Xilinx. System Generator funciona dentro del modelado y simulación del ambiente de Simulink. Su principal ventaja radica en la posibilidad de combinar bloques funcionales de Xilinx con Matlab y Simulink para crear un banco de pruebas y analizar así los datos generados por el modelo. El alto nivel de abstracción proporcionado por System Generator simplifica en gran medida el desarrollo de algoritmos y su verificación incluso en sistemas de comunicaciones multifrecuencia muy sofisticados.

Así, las librerías de Xilinx que incluyen bloques de comunicación, lógica de control, procesamiento de señales matemáticas y memoria, permiten integrar código HDL, funciones de Matlab y componentes de hardware diseñados para FPGAs de Xilinx, con el objetivo de crear modelos de sistemas de comunicaciones completos que se puedan simular en el entorno Simulink [23].

2.2.2. Fase 1. Selección de tecnologías y herramientas

Para el desarrollo del presente trabajo de Grado se seleccionó a nivel hardware el FPGA, debido a que la Universidad del Cauca cuenta con este dispositivo y como herramienta software a System Generator, porque en la fase 1, el sistema de comunicación se diseñó con esta tecnología, la cual no presenta compatibilidad con otros software de diseño.

- **Tecnología hardware**

La herramienta hardware reconfigurable seleccionada fue la tarjeta Starter Kit Board de Xilinx, la cual integra un FPGA de la familia Spartan 3A. Este dispositivo dispone de una serie de elemento lógicos, suficientes para la implementación de sistemas de complejidad media. A continuación, en la tabla 2.1 se indican los dispositivos que se encuentran dentro de esta familia con sus respectivas características.

Teniendo en cuenta lo anterior, el dispositivo en el cual se realiza la implementación del sistema de comunicaciones es el XC3S700A.

Tabla 2.1 Características de la familia Spartan 3A.

Dispositivo	compuertas del sistema	celdas lógicas equivalentes	matriz CLB				multiplicadores dedicados	Bits de bloque RAM	DCMs
			filas	columnas	CLBs	Slice			
XC3S50A	50K	1,584	16	12	176	704	3	54K	2
XC3S200A	200K	4,032	32	16	448	1,79	16	288K	4
XC3S400A	400K	8,064	40	24	896	3,58	20	360K	4
XC3S700A	700K	13,248	48	32	1,47	5,89	20	360K	8
XC3S1400A	1400K	25,344	72	40	2,82	11,3	32	576K	8

- **Tecnología software**

A nivel software la herramienta seleccionada corresponde al paquete de diseño, Ambiente de Software Integrado (ISE, *Integrated Software Environment*). Este software de Xilinx, permite analizar y compilar diseños HDL, analizar los tiempos de simulación, simular la reacción de un diseño a diferentes estímulos y configurar el hardware a través de una interfaz de programación [24]. De los componentes disponibles de este paquete, los que se utilizan para el presente trabajo de grado son:

System Generator: Es un Ambiente de Diseño Integrado (IDE, Integrated development environment), que hace uso de Simulink, como entorno de desarrollo. Tiene un flujo de diseño integrado, para pasar directo al archivo de configuración (*.bit) necesario para la programación del FPGA. Una característica importante es la generación automática del código del Lenguaje de Descripción Hardware para VHSIC (VHDL, *VHSIC Hardware Description Language*) y de un proyecto ISE del modelo que se desarrolle. Los bloques en XGS operan con valores booleanos o valor arbitrario en punto fijo para dar una mejor aproximación a la implementación hardware. Sin embargo, Simulink trabaja con números de punto flotante de doble precisión, por lo cual la conexión entre los bloques de System Generator y los de Simulink se realizan mediante los bloques “gateway”.

Project Navigator: Proporciona un análisis de todas las restricciones de diseño y el cálculo de todos los recursos hardware requeridos para una posible implementación.

Impact: Realiza la configuración directamente de los FPGAs Xilinx o Dispositivo Lógico Programable Complejo (CPLD, *Complex Programmable Logic Device*) y Memoria Programable de Solo Lectura (PROM, *Programmable Read-only Memory*) con diferentes interconexiones físicas de Xilinx como el cable serial, cable USB entre otros.

2.3. MODELADO

Con el fin de establecer las bases para el desarrollo apropiado del sistema de comunicaciones del presente trabajo de grado, de acuerdo a los requerimientos planteados en la sección 2.2.1, esta sección explora conceptualmente la obtención de un modelo básico para el sistema de comunicaciones banda base, con codificación BCH binaria con los esquemas de modulación MSK/FSK y un canal AWGN, analizando paso a paso el procesamiento que debe aplicarse a la señal proveniente de la fuente de información.

Esta selección de tecnologías se hace teniendo en cuenta que la universidad cuenta con la licencia del software matlab y con el uso del FPGA. Además que en la primera fase el sistema implementado se construyó con estas tecnologías.

2.3.1. Modelo de referencia

Teniendo en cuenta el sistema de comunicaciones de la figura 1.1, se elabora un modelo de referencia que cuenta con cinco componentes básicos y dos bloques adicionales: una fuente de información que genera la información k a transmitir; un codificador BCH binario que le agrega redundancia a ésta, transformándola en un bloque de palabra código n; un modulador MSK/FSK banda base que la adecua para ser transmitida; un canal de comunicaciones que adiciona ruido AWGN; un demodulador MSK/FSK banda base, que recupera la señal en una réplica lo más parecida posible a la emitida por la fuente de información; un decodificador BCH binario que corrige t errores en la palabra código n y obtiene en lo posible la información k y finalmente un componente encargado de evaluar los errores (BER/WER). La figura 2.3 permite observar los módulos que lo conforman.

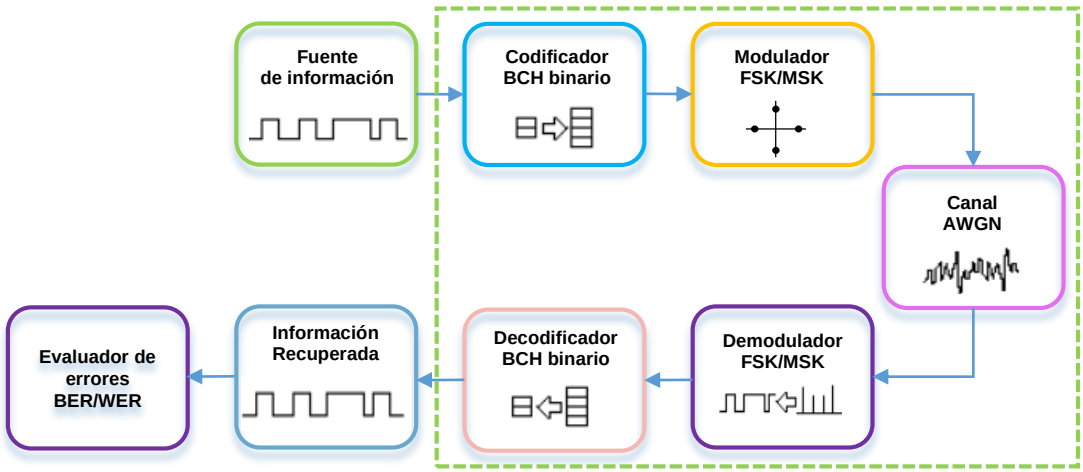


Figura 2.3 Modelo de referencia del sistema de comunicaciones.

2.3.2. Fase 2. Diseño del sistema

A partir del modelo de referencia, se diseñan los bloques correspondientes al codificador y decodificador. Para ello, se tiene en cuenta que los módulos del sistema de comunicaciones referentes a la fuente de información, los moduladores y demoduladores MSK/FSK y el canal AWGN fueron diseñados y construidos en el trabajo de grado "Análisis del desempeño de un sistema de comunicaciones con modulación MSK/FSK basado en hardware reconfigurable" [2], que se encuentra dentro del proyecto "Diseño e Implementación de un Prototipo de Comunicación de Datos Basado en Hardware Reconfigurable Fase 1" [25]. Además, se diseñan otros módulos para el análisis del desempeño a nivel de la BER/WER y para la visualización de la información obtenida al implementar el sistema.

Los parámetros generales que se tienen en cuenta para el diseño del codificador y decodificador se enuncian en la Tabla 2.2.

Tabla 2.2 Parámetros de diseño del codificador y decodificador BCH binario.

Campo de Galois	Códigos BCH binarios
$\boldsymbol{GF(2^3)}$	$\mathrm{BCH}(7,4)$
$\boldsymbol{GF(2^4)}$	$\mathrm{BCH}(15,5)$, $\mathrm{BCH}(15,7)$, $\mathrm{BCH}(15,11)$
$\boldsymbol{GF(2^5)}$	$\mathrm{BCH}(31,16)$, $\mathrm{BCH}(31,21)$ $\mathrm{BCH}(31,26)$

2.3.2.1. Codificador BCH binario

En la figura 2.4, se presenta el módulo del codificador BCH binario.

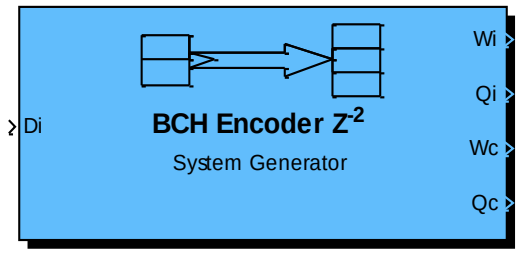


Figura 2.4 Bloque Codificador BCH binario *BCH Encoder.*

Este bloque, se encuentra conformado por las entradas y salidas, indicadas en la tabla 2.3.

Tabla 2.3 Entradas y salidas BCH Encoder System Generator.

TIPO	COMPONENTES	OPERACIÓN
Entradas	D_i	Bits de información de entrada.
Salidas	W_i	Palabra de información de tamaño k para codificar.
	Q_i	Bits de información D_i retrasados $2 * k$ bits para sincronizarla con las salidas del decodificador.
	W_C	Palabra código de tamaño n.
	Q_C	Bits serializados de la palabra código.

El bloque *BCH Encoder* está compuesto por dos subsistemas que realizan todo el proceso de codificación, los cuales se muestran en la Figura 2.5.

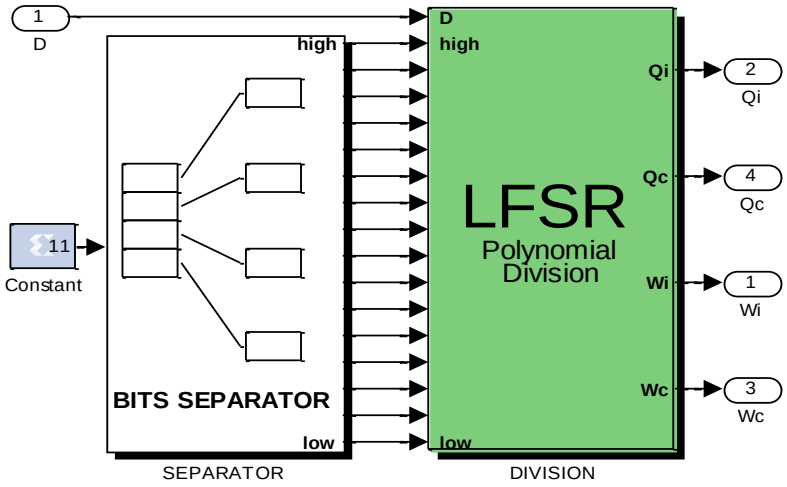


Figura 2.5 Subsistema del Codificador BCH Binario.

A continuación, se realiza una descripción de las funciones que tiene cada subsistema del codificador.

- **Constante, *Constant***

Tiene el valor del polinomio generador (variable *polynomial*), que cuenta con una precisión de 16 bits[12] como se ve en figura 2.6.

[12] La precisión es de 16 bits, debido a que la potencia máxima del polinomio generador es 16, correspondiente al código $BCH(31{,}26)$.

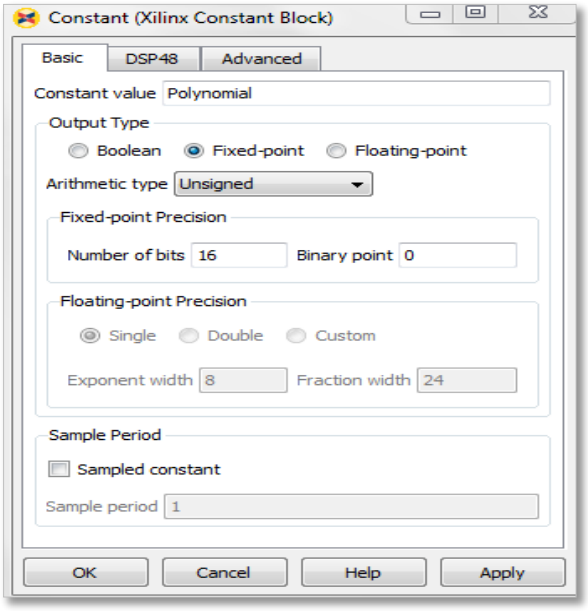


Figura 2.6 Configuración *Constant.*

- **Separador, *Separator***

Su función es separar el polinomio generador en cada uno de sus coeficientes y obtener una salida por cada uno. Para su construcción se requiere de un bloque *cast*, que toma el coeficiente menos significativo; para los siguientes, se hace un desplazamiento con los registros a una posición a la derecha, para obtener las otras salidas. En la figura 2.7 se observa el diseño del subsistema.

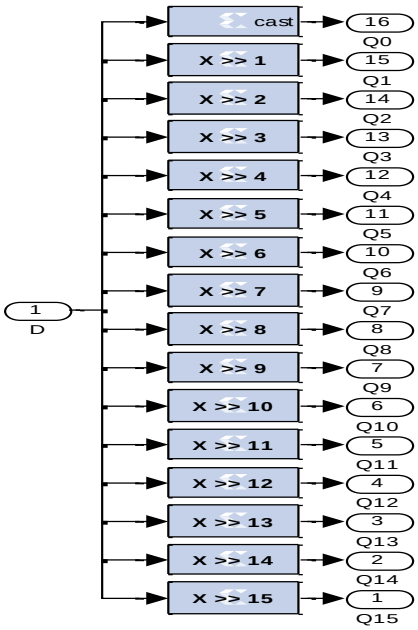


Figura 2.7 Estructura del subsistema *Separator*.

- **División, *Division***

Consta de tres partes básicas que tienen como función principal, obtener el cálculo de la palabra código, su diseño se basa en LFSR's con los cuales se realizan las divisiones polinomiales a nivel hardware dada por la ecuación 1.14. La figura 2.8 permite observar el esquema general del subsistema *división*.

En el anexo C, se aprecia el diseño para el proceso de la división polinomial con LFSR's [26].

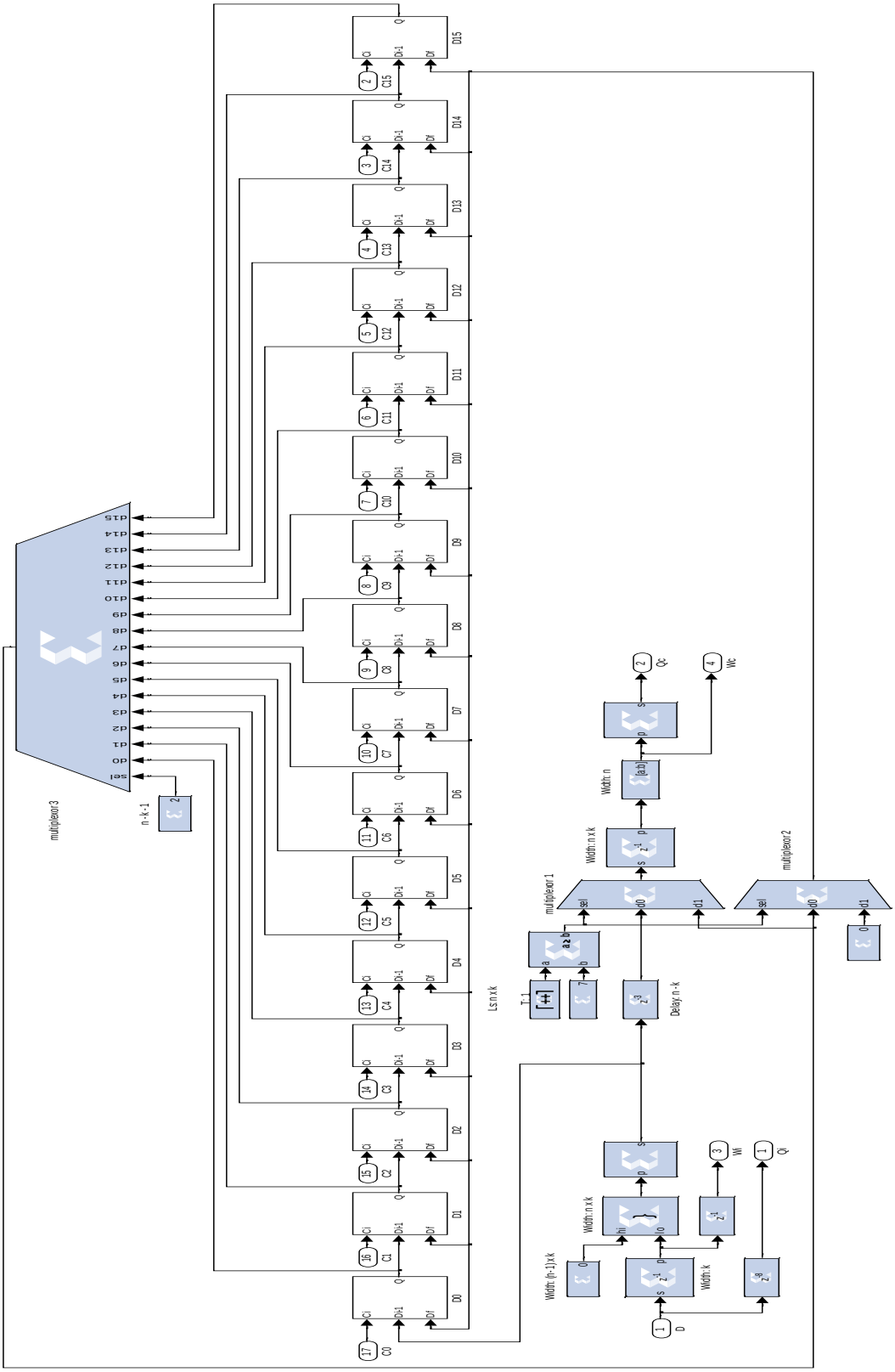


Figura 2.8 Componentes del subsistema *División.*

La primer parte se encarga del *concatenamiento* de los bits. Para su diseño se utiliza un registro serie paralelo, el cual genera las palabras de información de tamaño k y luego se concatena con una constante de valor cero y tamaño $(n-1)*n$, de esta manera se logra una palabra de $(n*k)$ bits que posteriormente se serializa para obtener una mayor frecuencia de bit de información. Esta primera parte, cuenta con dos salidas W_i y Q_i, las cuales se retrasan para ser sincronizadas con las salidas del codificador. Los módulos que componen esta parte se ilustran en la figura 2.9.

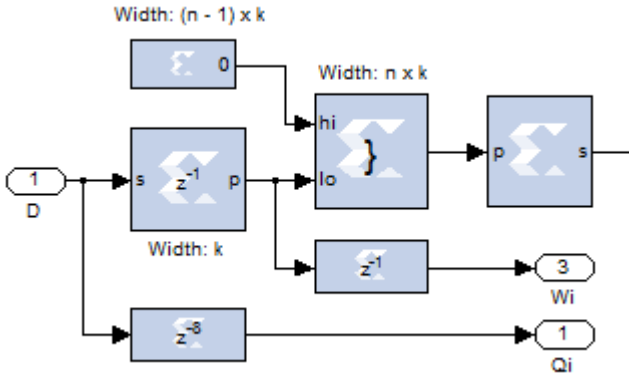


Figura 2.9 Concatenamiento, subsistema *División.*

La figura 2.10 presenta la segunda parte del subsistema *división*, el cual corresponde a la formación de la *palabra código*, que se hace a partir de elementos básicos de System Generator.

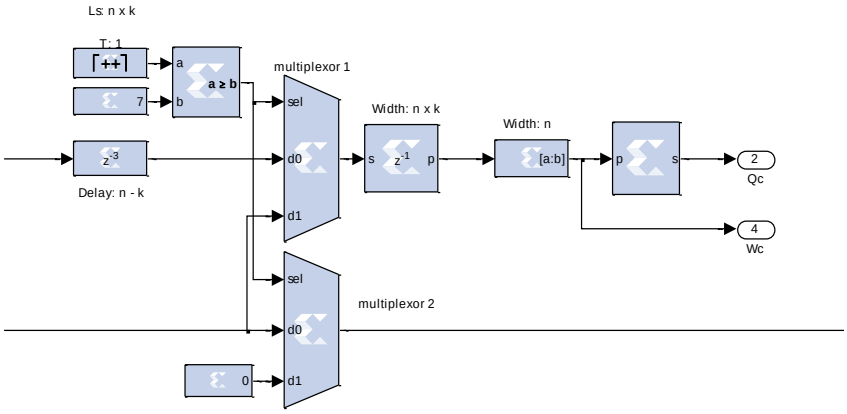


Figura 2.10 Formación palabra código, subsistema *División*.

El *multiplexor 1* tiene como función elegir entre la entrada de información serializada $(d0)$ o la de redundancia $(d1)$. La selección se hace a partir de la comparación entre un contador que va desde 0 hasta $(n*k)-1$ y de una constante de valor n; si el contador es menor al valor de la constante se obtiene a la salida del multiplexor 1 la

información serializada $(d0)$, por el contrario, si supera o iguala al valor de la constante, se obtiene a la salida la redundancia $(d1)$.

A la salida del *multiplexor 1*, se encuentra un registro serie paralelo, el cual concatena la redundancia con la palabra de información k; después por medio de un *slice*, se extrae la palabra código de la salida del registro serie paralelo. Por último esta información es enviada bit a bit al modulador.

El *multiplexor 2*, genera el bit de retroalimentación del sistema de LFSR (Df), para lo cual debe seleccionar entre la salida del *multiplexor 3* $(d0)$ y la constante cero $(d1)$, la cual es una entrada común para todos las etapas del LFSR, la elección se hace con la misma lógica del *multiplexor 1*. De esta manera, si el valor de la señal de selección es cero se obtiene la salida del *multiplexor 3* que corresponde a la salida del sistema, en caso contrario se opta por el valor cero de la constante, con el fin de que una vez ingresados los n bits, el sistema transmita el residuo de la división que se encuentra guardado en los registros de cada etapa del sistema LFSR, sin que se vea afectado por el polinomio generador.

La figura 2.11 corresponde a la construcción de la redundancia de la palabra código, haciendo uso de un *multiplexor* 3 y de sistemas LFSR.

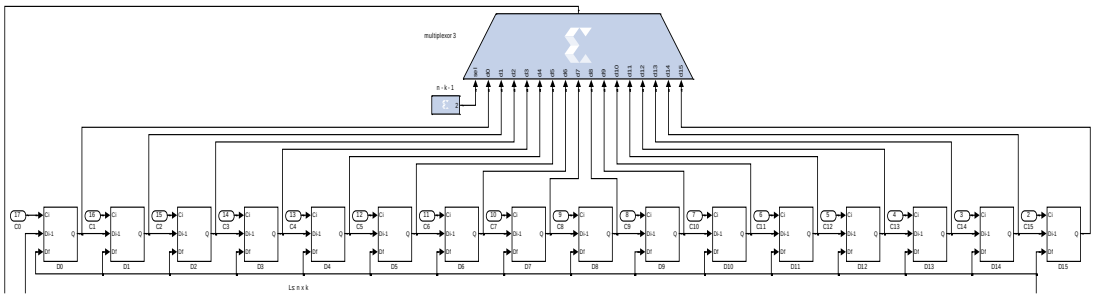


Figura 2.11 Redundancia del subsistema *División*.

Cada etapa del LFSR está compuesto por un *delay,* una compuerta *xor,* una compuerta *and* y por sus respectivas entradas y salidas como se ve en la figura 2.12.

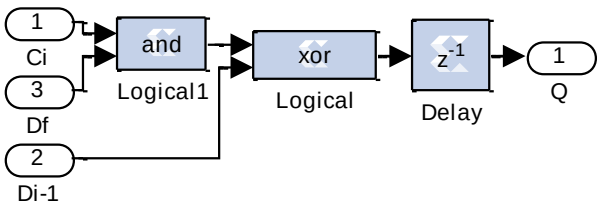


Figura 2.12 LFSR del subsistema *División*.

La *entrada* C_i corresponde a un coeficiente del polinomio generador; D_f es el bit de retroalimentación del polinomio y D_{i-1} la salida del sistema anterior. Si C_i es cero, significa que no existe el término en el polinomio, obteniendo a la salida el valor de la entrada D_{i-1}, debido a que la operación AND tiene un valor cero; por el contrario, si C_i vale uno, la compuerta XOR realiza la operación entre D_f y D_{i-1}.

El *multiplexor 3*, tiene como función determinar el punto de la salida del sistema, el cual depende del tipo de código BCH binario que fue configurado. La selección se hace por medio de una constante de valor $n-k-1$, que indica el número de etapas a usar de LFSR´s, obteniendo a la salida el resultado de la división, el cual llega al *multiplexor 2* y *multiplexor 1*.

2.3.2.2. Decodificador BCH binario

El diseño del decodificador BCH binario "BCH Decoder" se realiza con base al modelo matemático descrito en la sección 1.3.2. En la figura 2.13, se ilustra el bloque general, el cual contiene una serie de entradas que facilitan la llegada de los datos provenientes del demodulador y unas salidas que proporcionan la información decodificada.

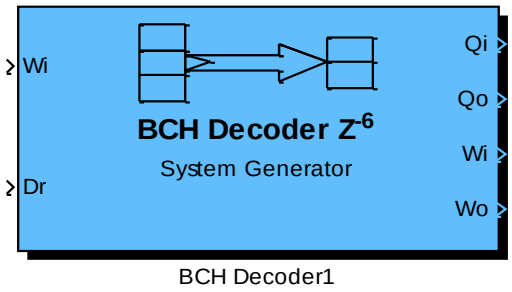


Figura 2.13 Bloque Decodificador BCH binario *BCH Decoder.*

De lo anterior, en la tabla 2.4 se definen tanto las entradas como las salidas del bloque decodificador.

Tabla 2.4 Entradas y Salidas del Bloque del Decodificador BCH binario.

TIPO	COMPONENTES	OPERACIÓN
Entradas	W_i	Palabra de información k original.
	D_r	*Stream* de bits a decodificar.
Salidas	Q_i	Bits de Información originales.
	Q_o	Bits de información decodificados.
	W_i	Palabra de información original k retrasada seis palabras de código.
	W_o	Palabra de información decodificada k.

El bloque *BCH Decoder*, está compuesto por varios bloques que conforma todo el proceso de decodificación; de este modo, se hace uso del bloque *Subsystem* de Xilinx para diseñar los seis subsistemas, indicados en la figura 2.14.

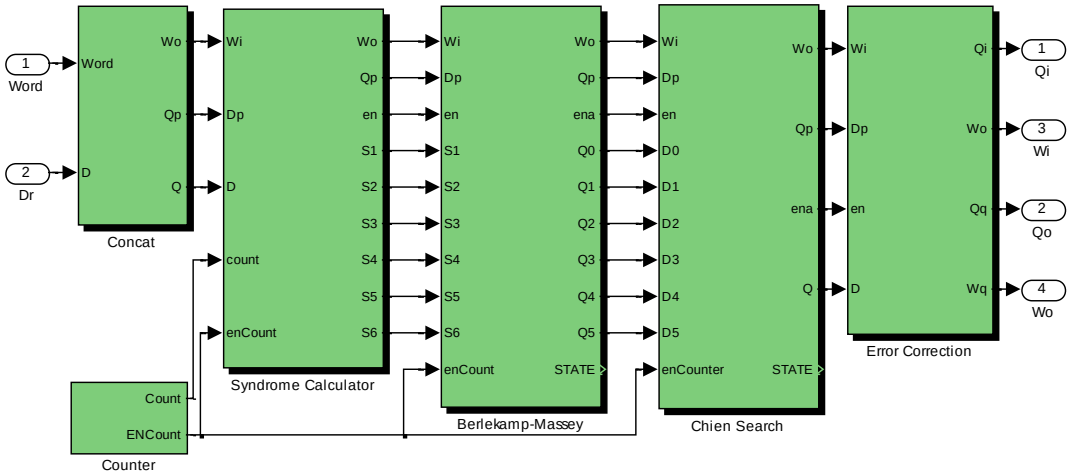


Figura 2.14 Subsistemas del bloque general del decodificador.

Cada subsistema ejecuta una función específica y realiza la operación a partir del cambio de una señal $enable$ del subsistema que lo precede. En seguida, se detalla cada subsistema.

- **Contador, *Counter***

Se utiliza para sincronizar los otros subsistemas; tiene dos salidas, la primera $Count$ contiene el valor del contador el cual tiene limitada la cuenta al valor $(n * k) - 1$; la segunda $ENCount,$ genera una señal en alto cada vez que el contador llegue al valor límite. En la figura 2.15 se observa los componentes de este subsistema.

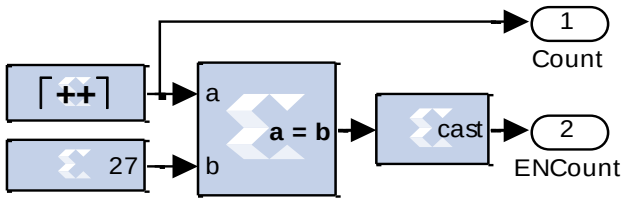


Figura 2.15 Componentes del subsistema *Counter*.

- **Concatenador, *Concat***

Este subsistema consta de dos entradas, la primera corresponde a $Word$ que es la misma W_i y la segunda es D_r. El bloque tiene la función de acumular los bits de D_r en

palabras de tamaño n para decodificar, por lo cual se utiliza un registro serie paralelo. A esta salida se le concatena una constante de valor 0 y tamaño $(k-1)*n$ y posteriormente se serializa con el fin de tener una mayor frecuencia en la salida Q; la salida Q_p representa la palabra a decodificar y W_o, la información original retrasada. La figura 2.16 muestra los elementos que conforman este subsistema.

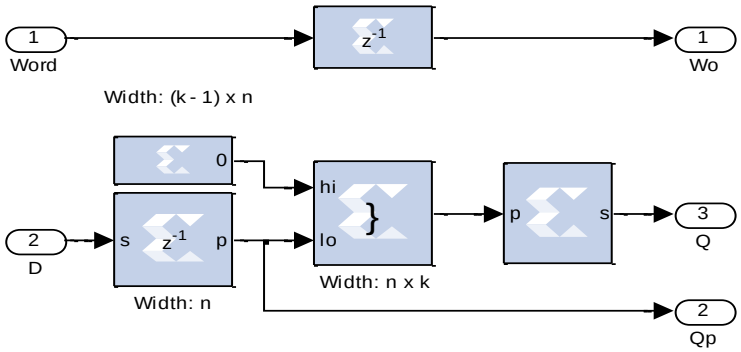


Figura 2.16 Componentes del subsistema *Concat.*

- **Calculador de Síndrome, *Syndrome Calculator***

Este subsistema calcula los síndromes del sistema descritos matemáticamente en la sección 1.3.2; los módulos que lo componen permiten realizar este cálculo, ilustrados en la figura 2.17 los cuales consta de cinco entradas descritas como:

- ✓ W_i, palabra de información.
- ✓ D_p, salida Q_p.
- ✓ D, bits a decodificar.
- ✓ $Count$, cuenta del contador de control.
- ✓ $ENCount$, señal habilitadora del contador.

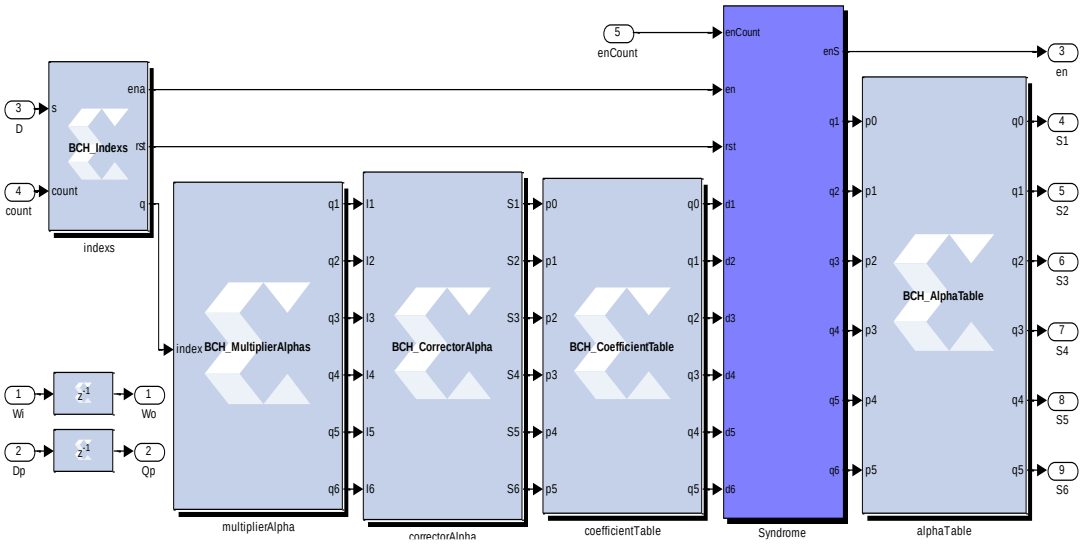


Figura 2.17 Componentes del subsistema *Syndrome Calculator.*

BCH_Indexs: Entrega el valor de la potencia[13] de cada uno de los términos del polinomio que representa la palabra de entrada s con valores de 0 hasta $n-1$, teniendo en cuenta que el último valor n representa la ausencia del término. Este tiene tres salidas, que corresponden respectivamente a $enable$, utilizada para desbloquear el subsistema *Syndrome*, $reset$ que borra los valores de los registros que guardan los síndromes y Q tiene los valores de las potencias.

MultiplierAlpha: Calcula la potencia de cada término del polinomio al ser evaluado el α correspondiente al síndrome. Para los códigos a implementar las raíces tienen potencias de 1 a $2t$. El resultado del cálculo anterior debe ser normalizado, por lo cual se realiza la operación $mod n$; debido a que el valor de n no es potencia de 2 y no es posible implementarla en el FPGA de forma directa, con lo cual se hace operaciones con el valor $n+1$. Primero, se realiza una división entre el termino calculado y $n+1$; segundo, se calcula el resto entre el término y $n+1$, finalmente se calcula la suma de los resultados anteriores.

Por ejemplo, para el cálculo del síndrome dos s_2, con el polinomio recibido $r(x) = x^6 + x^2$ en un $GF(2^3)$, se tiene:

$$s_2 = r(\alpha^2) = (\alpha^2)^6 + (\alpha^2)^2 = \alpha^{12} + \alpha^4.$$

Para determinar la potencia del primer término se realiza el siguiente procedimiento:

[13] Para el subsistema BCH_Indexs, se trabaja a nivel de potencias, por lo cual al polinomio $r(x)$, se le debe extraer sus potencias, por ejemplo el polinomio $r(x) = x^6 + x^2$, las potencias son respectivamente 6 y 2.

1. División

$$\frac{potencia\ de\ \alpha}{n+1} = \frac{12}{8} = 1.$$

2. Cálculo del resto

$$12\ mod\ 8 = 4.$$

3. Cálculo de la suma

$$4 + 1 = 5.$$

Obteniendo,

$$S_2 = r(\alpha^2) = (\alpha^2)^6 = \alpha^{12} = \alpha^5$$

CorrectorAlpha: Su función es corregir el valor de la potencia de α cuando el valor calculado en el subsistema anterior es superior a n. Entonces si es mayor a n, se resta este resultado, por el contrario se mantiene fijo.

CoefficientTable: Este módulo contiene un vector con el valor binario, correspondiente a la potencia α normalizada; este es creado según el campo de Galois al cual corresponde el código. Las tabla A.2 y A.3 del anexo A muestran la representaciones binarias de los α para cada campo de Galois. Por ejemplo, la representación binaria de α^5 en $GF(2^3)$ es 1111.

Syndrome: Una vez se genera cada término del síndrome, este módulo, genera la suma de los valores α para cada uno de los síndromes. La figura 2.18 presenta su estructura.

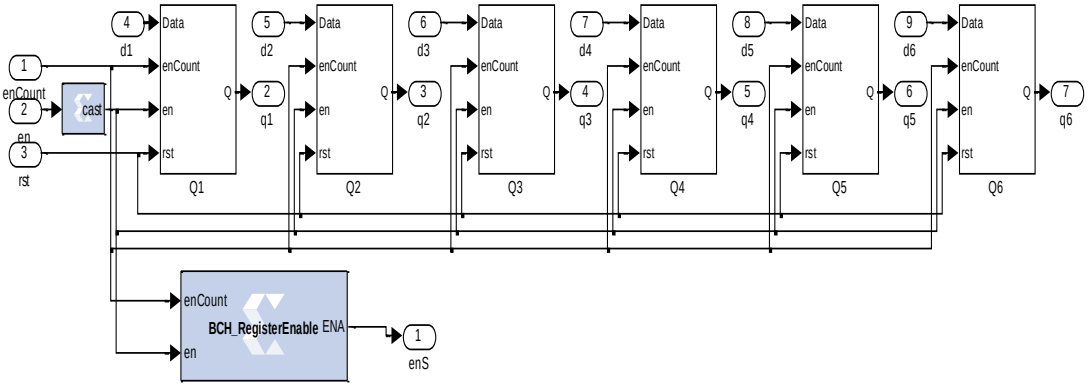


Figura 2.18 Componentes del módulo *Syndrome.*

Se tiene seis bloques sumadores Q, uno para cada bit. En la figura 2.19 se presenta la estructura de cada uno; *XOR* realiza la suma $mod2$ que luego es almacenada en el registro que posteriormente se utiliza en la suma del siguiente término, éste también tiene una entrada $reset$ que se utiliza para limpiar el registro cada vez que se hace

una nueva suma. *BCH_RegisterSyndrome* de la figura 2.19, tiene la función de guardar el resultado de la adición en el momento que la señal en se activa y finalmente cuando $enCount$ se activa, el síndrome calculado es colocado a la salida Q.

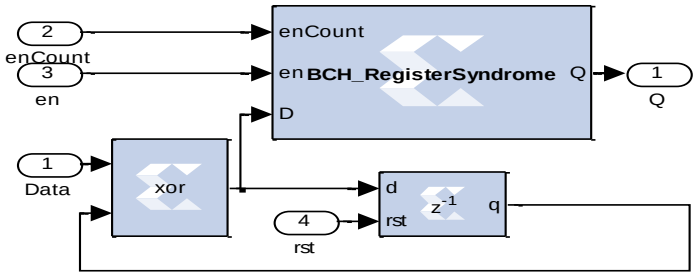


Figura 2.19 Componente del bloque sumador Q.

El *BCH_Register Enable* de la figura 2.18, activa la señal $enable$ del siguiente subsistema que corresponde al calculador Berlekamp-Massey; cuando en se activa, ésta espera a que $enCount$ se active para colocar la señal ENA en alto y enviar el resultado por la salida enS, indicando que la etapa de calculador de síndrome tiene un nuevo valor calculado.

Alpha table: Realiza la función inversa al subsistema *CoefficientTable*, es decir toma el valor binario y lo convierte en una potencia de α.

- **Berlekamp-Massey**

Este subsistema realiza matemáticamente el algoritmo dado por el diagrama de flujo ilustrado en la figura B.1 del apéndice B, para calcular el polinomio localizador de error $\Lambda(x)$; su construcción sigue todo el proceso de este algoritmo pero no se implementa con LFSR´s sino con máquinas de estado, debido a que con LFSR´s se requiere un modelo para cada tipo de código incrementando los recursos hardware. La figura 2.20 presenta los módulos que lo conforman.

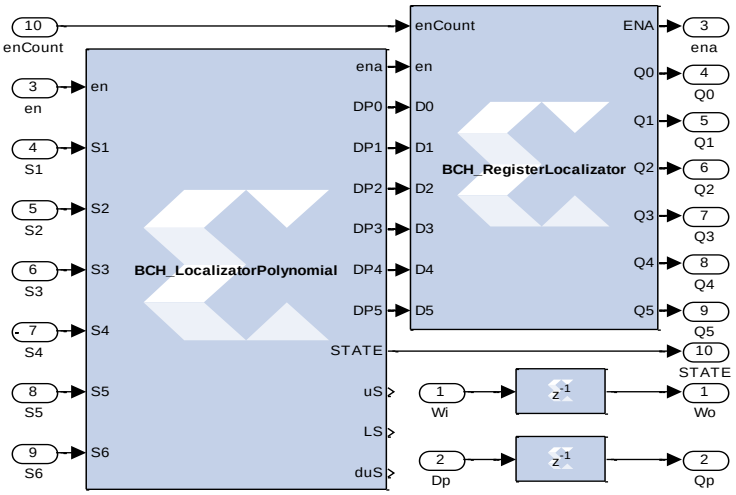


Figura 2.20 Composición del subsistema Berlekamp Massey.

BCH_LocalizatorPolinomial: Implementa una máquina de estados. El cual tiene todas las funciones para realizar el algoritmo completo. En la figura B.2 del apéndice B se aprecia su lógica.

BCH_RegisterLocalizator: Guarda el resultado del polinomio localizador de error y lo mantiene sincronizado con la señal $enCount$.

- **Chien Search**

Este subsistema tiene la función de calcular la posición de errores en la palabra a decodificar. Contiene dos módulos ilustrados en la figura 2.21. El primero, el *BCH_chienSearch* se encarga de encontrar las posiciones de error, realizar el cálculo del inverso de cada posición y construir el polinomio patrón de error y el segundo, *BCH_RegisterChien* lo guarda al llegar a la entrada D, cuando se habilita la señal en y lo envía por la salida Q una vez $enCount$ se habilite. En la figura B.3 del apéndice B se muestra la lógica de este subsistema mediante máquinas de estado.

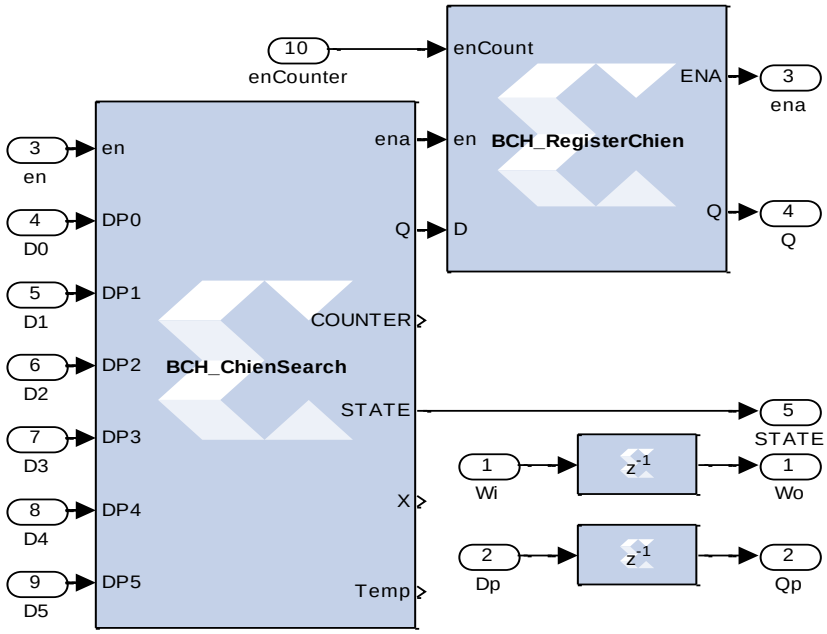


Figura 2.21 Componentes del subsistema *Chien Search.*

- **Error Correction**

Este subsistema suma la palabra recibida D_p al patrón de error D y entrega la información k decodificada cuando en este habilitada. La información se entrega en dos formatos, la palabra de tamaño k en la salida W_q y los bits serializados en la salida Q_q. La figura 2.22, presenta los componentes de este subsistema.

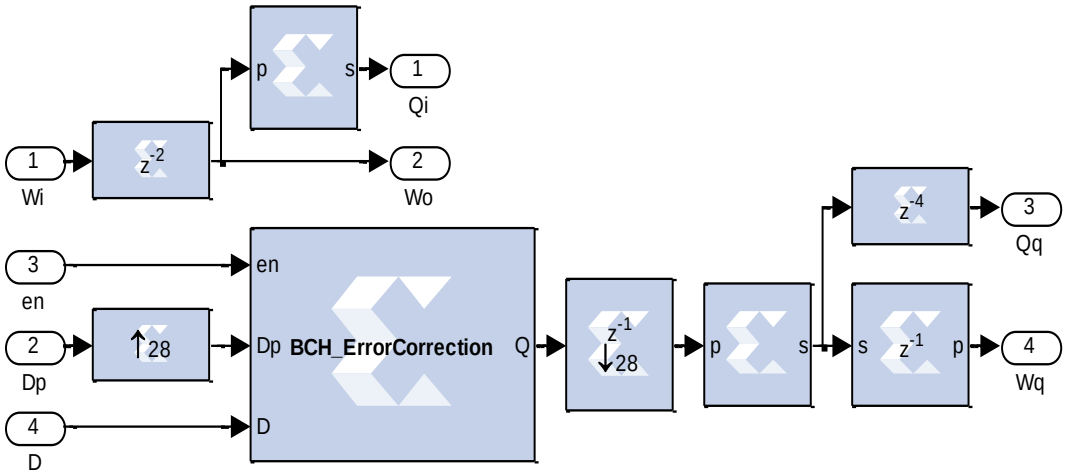


Figura 2.22 Componentes del subsistema *Error Correction.*

2.3.2.3. Bloques Adicionales del sistema de comunicaciones

- **Sincronizador, *Sincronizator***

Para alcanzar la sincronía entre los bloques de modulación MSK/FSK de la fase 1, con los bloques de codificación BCH binario; se diseña el bloque *sincronizator*, que se encarga de generar un retardo a las señales de entrada con el fin de sincronizarlas. En la figura 2.23 se evidencia el subsistema con sus componentes.

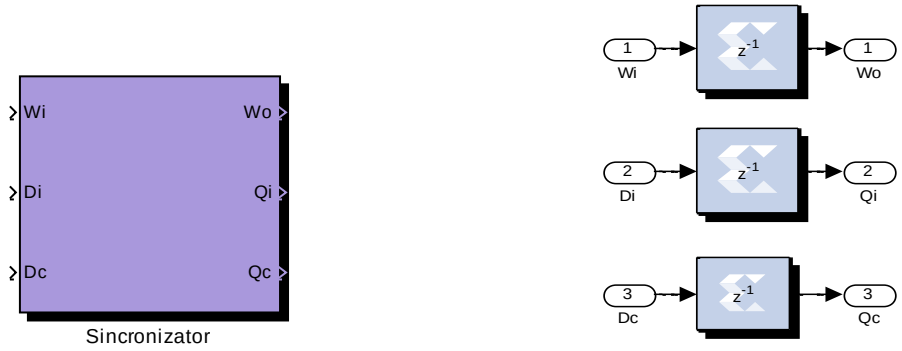


Figura 2.23 Componentes del *Sincronizator:* a) Subsistema y b) Estructura.

- **Calculador de BER/WER, *BER/WER Calculator***

Este subsistema tiene la función de calcular la cantidad de errores producidos a nivel de bit y de palabra en el sistema, dentro de una determinada cantidad de bits transmitidos como se observa en la figura 2.24.

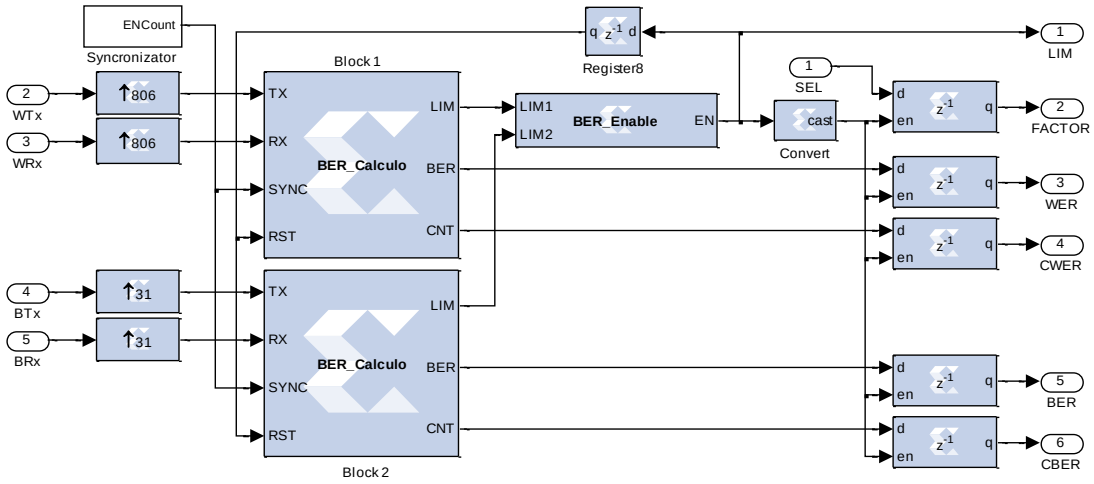


Figura 2.24 Componentes del Subsistema *BER/WER Calculator.*

A continuación, se describe cada uno de sus componentes.

***BER_Calculo**,* consta de dos bloques denominados *BLOCK 1* y *BLOCK* 2 que tiene la función de calcular las palabras y bits erróneos respectivamente; el primero cuenta con las entradas WTx, que son las palabras transmitidas originalmente y WRx que son las decodificadas; el segundo tiene dos entradas BTx, que son los bits transmitidos originalmente y BRx los decodificados; estos dos bloques tienen en común la entrada *SYNC,* encargada de informar cuando se finaliza el conteo de palabras o de bits observados y *RST*, que reinicia los valores de las variables cada vez que exista nueva información. En la figura B.4 del apéndice B, se describe la lógica de este bloque.

***BER_Enable**,* este bloque consta de dos entradas *LIM1* y *LIM2*. La primera indica la finalización de cuenta de las palabras erradas, la cual proviene de *BLOCK 1* y la segunda la de la cuenta de bits errados proveniente del *BLOCK 2*, al aparecer esta señal de una de las cuentas, se habilita *EN*, la cual se encarga de habilitar los registros para que guarden los valores de las salidas de los bloques de *BER Calculador*, información que posteriormente es entregada al bloque *Serial Transmition*.

- **Sistema de Control, *Control system***

Genera los valores de la desviación estándar de ruido (σ) para el canal AWGN del sistema de comunicaciones y el (E_b/N_O). La figura 2.25 presenta los módulos que lo conforman.

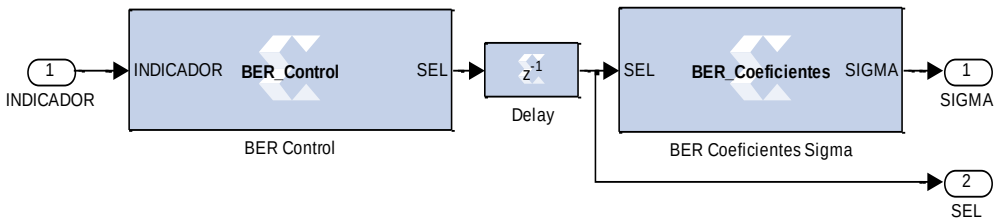


Figura 2.25 Subsistema *Control System*.

A continuación, se describe cada módulo.

BER_Control, encargado de variar el valor de E_b/N_O, desde 0 al máximo valor configurado al sistema.

***BER_Coeficientes**,* selecciona el valor σ de un vector constante, a partir del E_b/N_O en la entrada *SEL*, los valores de desviación se calculan a partir de la ecuación 2.1.

$$\sigma = \frac{1}{\sqrt{2 \times (E_b/N_0)}} \tag{2.1}$$

- **Transmisión serial, *Serial Transmission***

Este subsistema transmite serialmente los datos obtenidos del bloque *BER/WER Calculator*, hacia la aplicación de escritorio *Serial* Communication, diseñada para hacer el cálculo de la BER/WER que permite observar los valores de una forma amigable y rápida al usuario. En la figura 2.26 se observan los módulos que conforman el sistema de transmisión serial.

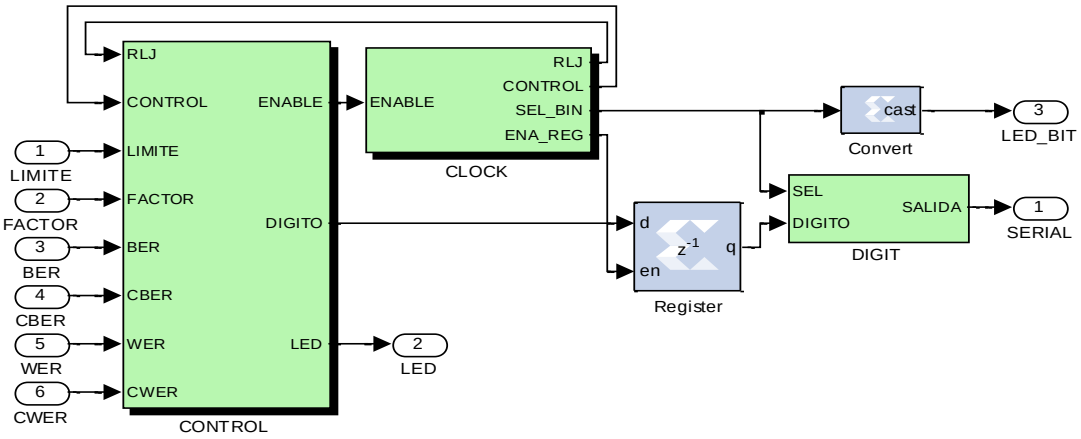


Figura 2.26 Componentes del subsistema *Serial Transmission*.

La figura 2.27 corresponde a los bloques que hacen parte del módulo *CONTROL*, los cuales tienen la función de construir la trama que contiene la información a enviar de forma serial; este módulo entrega a la salida el byte que se desea escribir en el puerto serial.

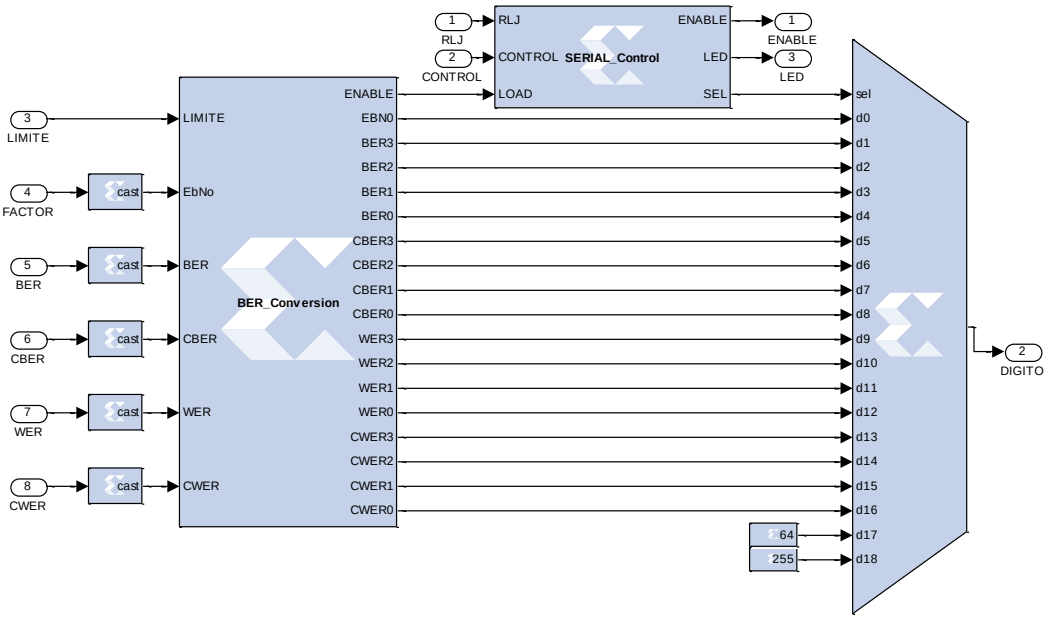


Figura 2.27 Bloques que conforman al módulo *CONTROL*.

En seguida se realiza una descripción de las funciones que tiene a cargo cada uno de los bloques.

Cast, realizan una casteo[14] de las entradas en enteros de 32 bits.

BER_Conversion, este bloque segmenta todos las entradas en bytes para los valores de *BER, CBER, WER, CWER* que son divididos en cuatro bytes cada uno, teniendo como el byte menos significativo el grupo de bits de 0 a 7, el siguiente de 8 a 15 y así sucesivamente hasta completar los 32 bits; en cambio el valor *FACTOR* es un byte y corresponde al valor de E_b/N_0; *BER,* los bits errados; *CBER,* la cantidad de bits transmitidos; *WER,* las palabras erradas y *CWER* la cantidad de palabras transmitidas.

SERIAL_Control, organiza el formato de envío de la información. Consta de dos entradas que provienen del bloque externo *CLOCK*; la primera de ellas *RLJ* es la señal de reloj del bit de transmisión serial que se utiliza para sincronizar el envío de la información por el puerto serial, la segunda, CONTROL indica en qué momento termina la transmisión de un byte por el canal serial, con el fin de seleccionar el siguiente byte a transmitir; por último la señal *Load* proveniente de *BER_CONVERSION* indica cuando hay nueva información para transmitir. Las salidas corresponden a *ENABLE,* que habilita o deshabilita el funcionamiento del bloque externo CLOCK; *LED,* el cual indica en qué momento se transmite una nueva

[14] Casteo: Convierte cada muestra de entrada a un número de un tipo aritmético deseado.

información y *SEL*, que selecciona del multiplexor el byte a transmitir por el puerto serial. En la figura B.5 del apéndice B, se describe la lógica de este bloque.

MULTIPLEXOR, selecciona el byte que se va a transmitir por la salida *Digito*.
La figura 2.28 ilustra la estructura del módulo CLOCK, el cual se encarga de generar las frecuencias en baudios, para su posterior transmisión por el puerto serial del FPGA.

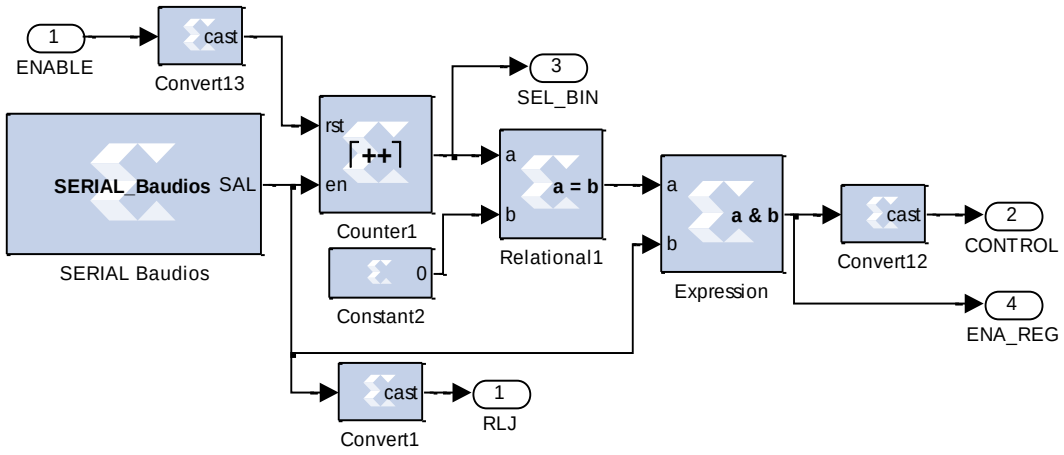


Figura 2.28 Bloques que conforman al módulo CLOCK.

Serial_Baudios, genera la señal de reloj para la transmisión de información por el puerto serial, utiliza un contador que va desde cero hasta el valor de una constante *Cuenta*[15]; cada vez que el valor de cuenta es cero coloca la señal de salida en uno, el valor de *Cuenta* determina la frecuencia de transmisión de la información.

Counter1, controla la transmisión de los bits de un byte, además de los bits de Start y Stop propios de la protocolo de transmisión serial. Tiene dos entradas, *rst* y *ena*; *rst* activo mantiene el contador con el valor cero, cuando es inactivo cada vez que *ena* se active realiza un incremento de la cuenta la cual es reflejada en la salida *SEL_BIN,* esta salida es utilizada en el bloque externo *DIGIT*.

Relational1, coloca a su salida un valor en alto cada vez que el contador tiene el valor cero.

Expression, genera una señal en alto cada vez que se va a transmitir un byte, por medio de una compuerta *AND* entre *Relational1* y *Serial_Baudios,* siempre y cuando ambas lo estén. Tiene dos salidas, la primera *Control* que se utiliza en el bloque

[15] Cuenta: Se utiliza para el cálculo de la frecuencia en baudios, para el caso de 9600 baudios cuenta es el resultado de dividir $50mHz/9600$.

externo *CONTROL*; la otra *ena_reg*, utilizada como señal de *enable* en el bloque externo *Register*.

***Register*,** guarda la señal que proviene de la salida *DIGITO* del bloque *CONTROL,* cuando *ena_reg=1*.

En la figura 2.29 se ilustra el módulo *DIGIT,* que transmite el byte de forma serial. Consta de un multiplexor de 11 entradas y una señal de selección *SEL;* cuando *SEL* vale uno transmite el bit de Start el cual corresponde al valor cero, después comienza a transmitir bit a bit el byte desde el menos al más significativo y por último el bit de Stop; de esta manera D1 es el bit Start, D2 a D9 es la información, D10 transmite el bit de Stop y D0 corresponde a la línea en reposo o *Idle*.

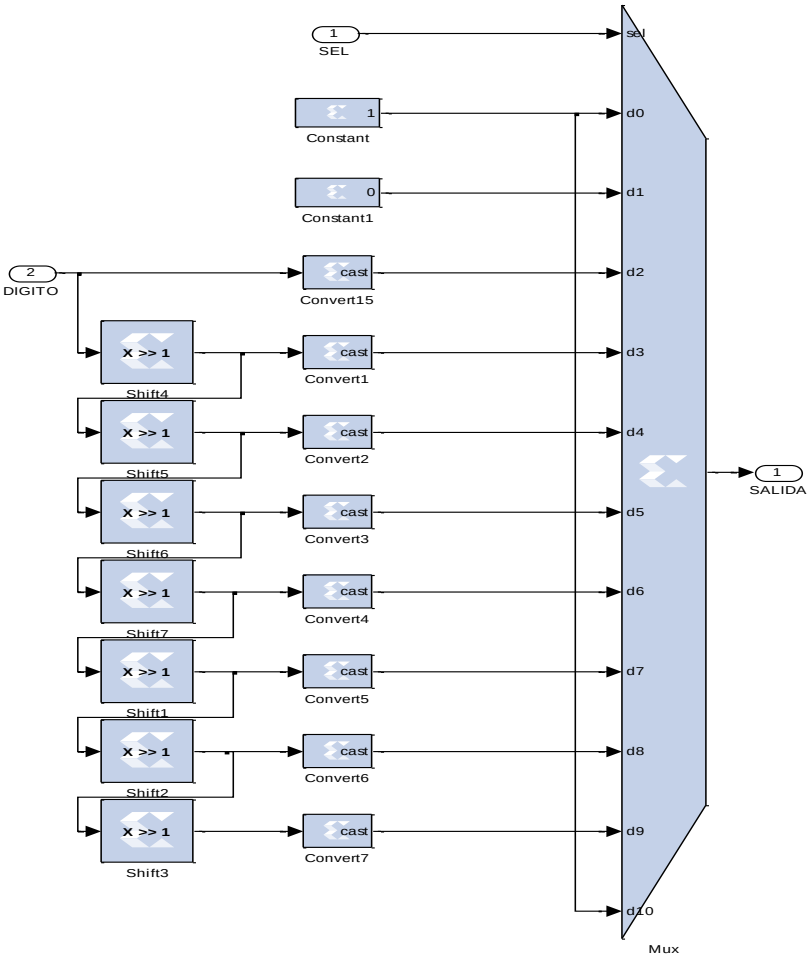


Figura 2.29 Módulos que conforman a *DIGIT*.

Serial Communication

Es una aplicación de escritorio diseñada en C Sharp[16] que realiza el cálculo de la *BER* y la *WER* a partir de la información enviada de forma serial por el FPGA. Calcula el valor de BER/WER a partir de las ecuaciones 2.2 y 2.3.

$$BER = \frac{número\ de\ bit\ errados}{bit\ transmitidos}, \tag{2.2}$$

$$WER = \frac{número\ de\ palabras\ erradas}{palabras\ transmitidas}. \tag{2.3}$$

Una vez generados estos valores, *Serial Communication* ejecuta varios métodos encargados de capturar y visualizar la BER/WER. En la figura 2.30 se muestra las funcionalidades de la aplicación por medio del modelo de Casos de Uso[17].

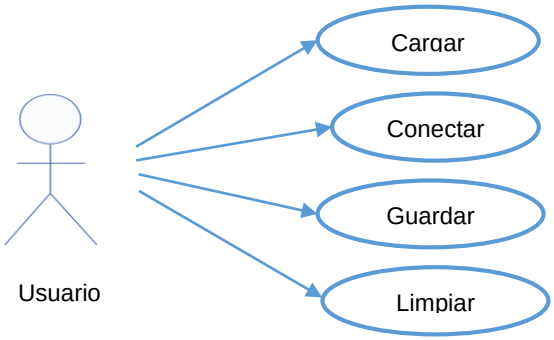


Figura 2.30 Casos de Uso de *Serial Communication*.

Descripción de los casos de uso.

En las tablas 2.5, 2,6, 2.7 y 2.8 se indica la descripción de los Casos de Uso.

Tabla 2.5 Caso de uso "Cargar".

Nombre	**Cargar**
Actor	Usuario.
Propósito	Permite cargar los puertos COM habilitados del computador.
Resumen	Una vez el usuario ejecute la aplicación, debe presionar el botón *"reload"*, para conectar el puerto serial del FPGA con el puerto COM del computador.

[16] C Sharp: Lenguaje de programación diseñado para compilar diversas aplicaciones que se ejecutan en .NET Framework. Es simple, eficaz, con seguridad de tipos y orientado a objetos.
[17] Caso de uso: Representa la forma de interacción entre un usuario y el sistema desarrollado.

Tabla 2.6 Caso de uso "Conectar".

Nombre	**Conectar**
Actor	Usuario.
Propósito	Habilitar la conexión entre el FPGA y el computador.
Resumen	Una vez se hace el reconocimiento de los puertos, se selecciona el puerto COM a utilizar y se presiona el botón *"connect"* que permite la visualización de información entregada por el dispositivo. En caso que el usuario desea terminar la conexión, nuevamente presiona el mismo botón y pasa a "*disconnect"* inmediatamente.

Tabla 2.7 Caso de uso "Guardar".

Nombre	**Guardar**
Actor	Usuario.
Propósito	Almacenar los datos en un archivo .csv.
Resumen	Cuando la información se visualiza, el usuario en cualquier momento puede presionar el botón *"save"* para que ésta se guarde en un archivo .csv.

Tabla 2.8 Caso de uso "Limpiar".

Nombre	**Limpiar**
Actor	Usuario.
Propósito	Capturar nueva información.
Resumen	Si el usuario desea descartar lo que se visualiza, puede presionar el botón *"clear"* que limpia la ventana de visualización.

2.3.2.4. Diseño total del sistema de comunicaciones

Posteriormente el diseño de los bloques anteriormente descritos se integra con el modelo construido en la fase 1. En la figura 2.31, se puede visualizar todos los componentes del sistema de comunicaciones con la modulación FSK.

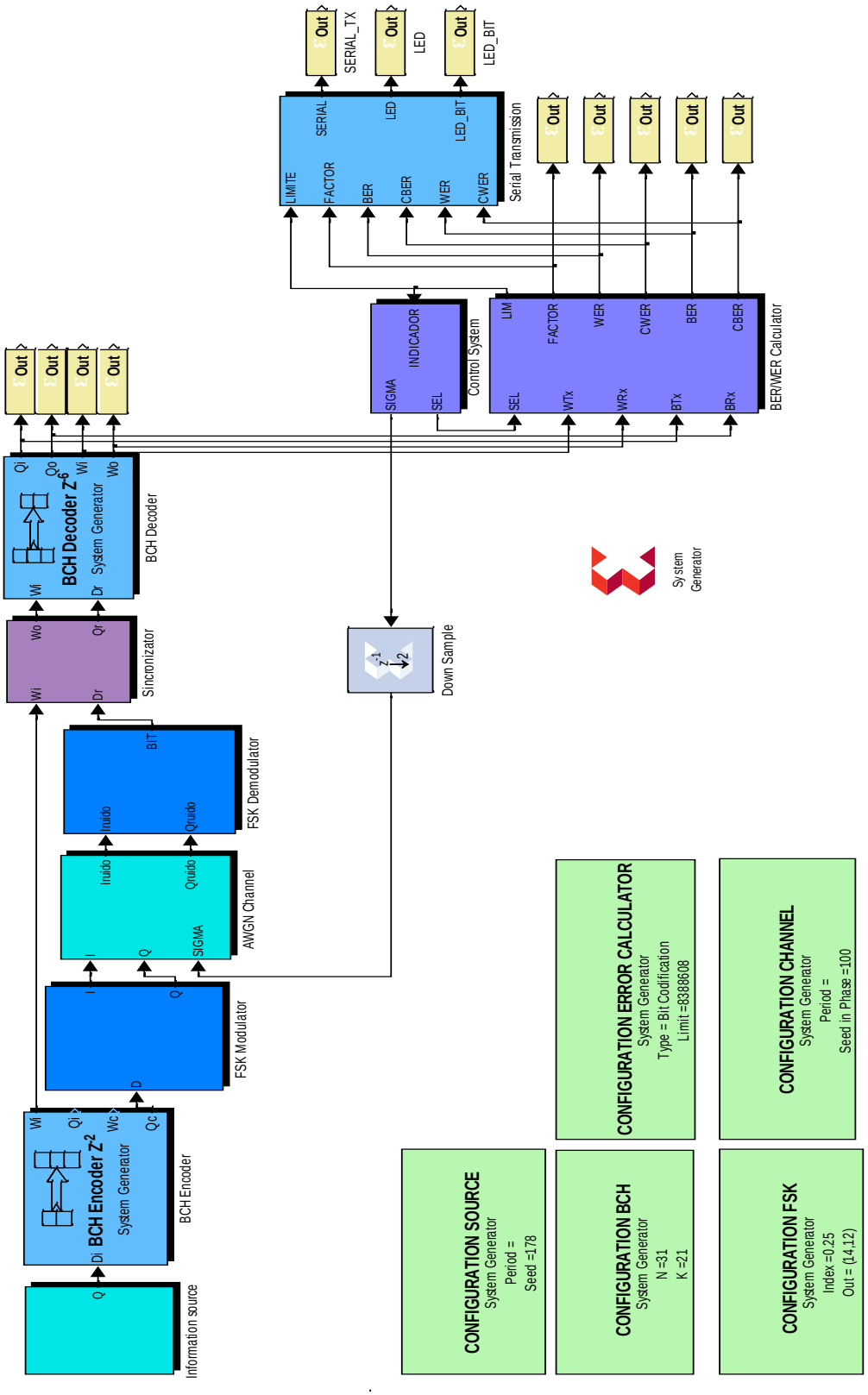


Figura 2.31 Diseño total del sistema de comunicaciones.

Para las configuraciones de los bloques que conforman el sistema se tiene las siguientes interfaces:

- **CONFIGURATION SOURCE**

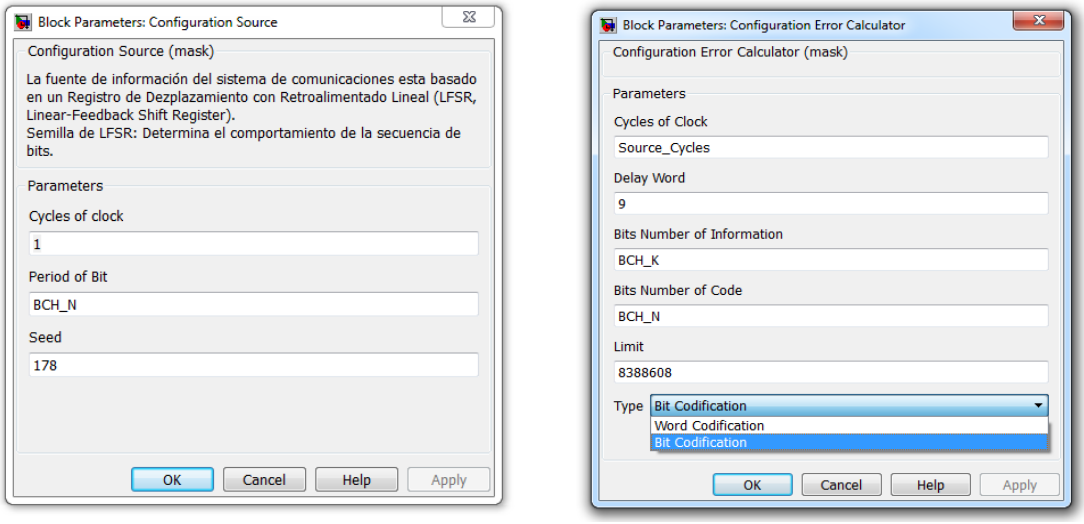
  - ✓ ***Cycles of clock:*** Especifica los ciclos del reloj de un bit.
  - ✓ ***Period of bit:*** Está asociado al tipo de codificación.
  - ✓ ***Seed:*** Semilla aleatoria del generador.

Un ejemplo de la configuración se puede ver en la figura 2.32 (a).

- **CONFIGURATION ERROR CALCULATOR**

  - ✓ ***Delay Word:*** Corresponde a los retardos del sistema total. Tiene un valor de nueve (9); es la suma de los dos retardos producidos por el codificador, seis dados por el decodificador y uno dado por los esquemas de modulación.
  - ✓ ***Bits Number of Information:*** Parámetro que indica la información dada en bits.
  - ✓ ***Bits Number of Code:*** Parámetro que indica la palabra código n.
  - ✓ ***Limit:*** Indica el valor de información a transmitir. El valor máximo corresponde a 2^{32}.
  - ✓ ***Type:*** Indica el tipo de información que se va a enviar. Puede ser en bits o palabras de tamaño de k bits. Por ejemplo, si se coloca el valor de 1000 en Limit y se selecciona *Word codificación* significa que se van a enviar 1000 palabras y por tanto $1000 * \mathrm{k}$ bits o por el contrario se selecciona *bit codificación*; la información enviada corresponde a 1000 bits y $1000/\mathrm{k}$ palabras.

Un ejemplo de la configuración se puede ver en la figura 2.32 (b).



a) b)

Figura 2.32 Configuración de parámetros: a) Configuration Source y b) Configuration Error Calculator.

- **CONFIGURATION CHANNEL**

  - ✓ **Period:** Corresponde a los ciclos del reloj.

  - ✓ **Seed in Phase:** Corresponde al valor de semilla que tendrá el generador de ruido blanco gaussiano en la componente en fase (I).

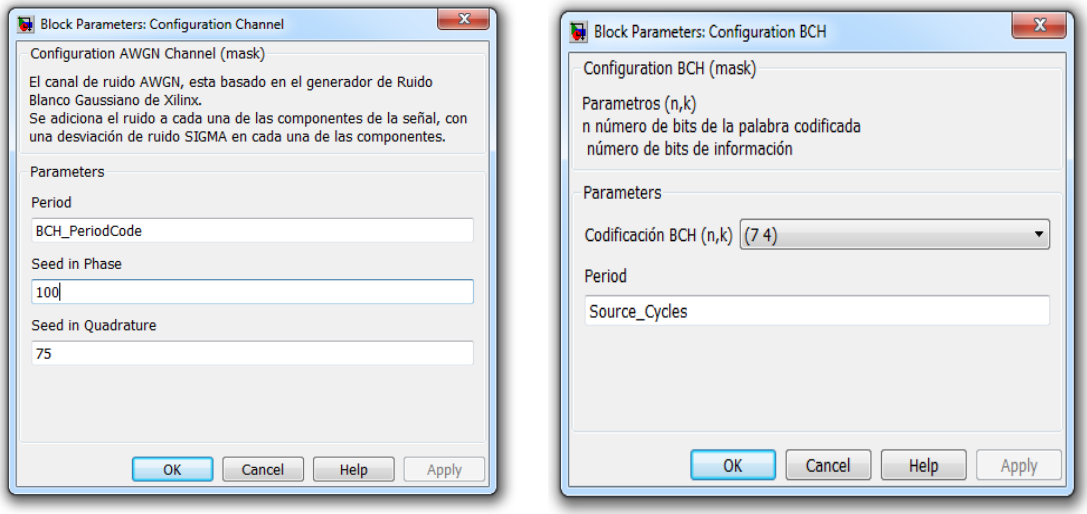
  - ✓ **Seed in Quadrature:** Corresponde al valor de semilla que tendrá el generador de ruido blanco gaussiano en la componente en cuadratura (Q).

Un ejemplo de la configuración se puede ver en la figura 2.33 (a).

- **CONFIGURATION BCH**

  - ✓ **Codificación $\mathbf{BCH}(\boldsymbol{n}, \boldsymbol{k})$:** Fija el tipo de codificación.

Un ejemplo de la configuración se puede ver en la figura 2.33 (b).



a) b)

Figura 2.33 Configuración de parámetros: a) Configuration AWGN Channel y b) Configuration BCH.

- **CONFIGURATION FSK**

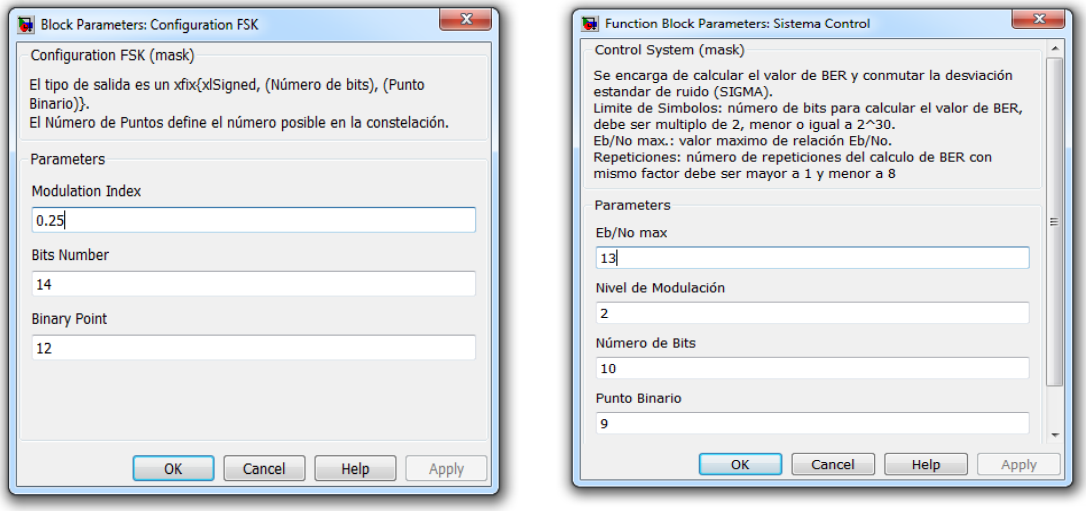
  - ✓ **Modulation Index:** Indica el índice de modulación h.
  - ✓ **Point Number:** Determina el número de puntos posibles en la constelación, es un valor potencia de 2.
  - ✓ **Bits Number and Binary Point:** Fija la longitud binaria de las componentes I/Q.

Un ejemplo de la configuración se puede ver en la figura 2.34 (a).

- **CONFIGURATION SYSTEM CONTROL**

  - ✓ $\boldsymbol{E_b/N_o}$ **Max:** Corresponde al nivel de E_b/N_o máximo.
  - ✓ **Nivel de modulación:** Corresponde al nivel de modulación de FSK/MSK utilizados.
  - ✓ **Numero de bits y Punto binario:** Fija la precisión de salida de este bloque.

Un ejemplo de la configuración se puede ver en la figura 2.34 (b).



a) b)

Figura 2.34 Configuración de parámetros: a) Configuration FSK y b) Configuration System Control.

2.4. SIMULACIÓN

2.4.1 Fase 3. Simulación del sistema

La Metodología de simulación se ilustra en la figura 2.35 [27].

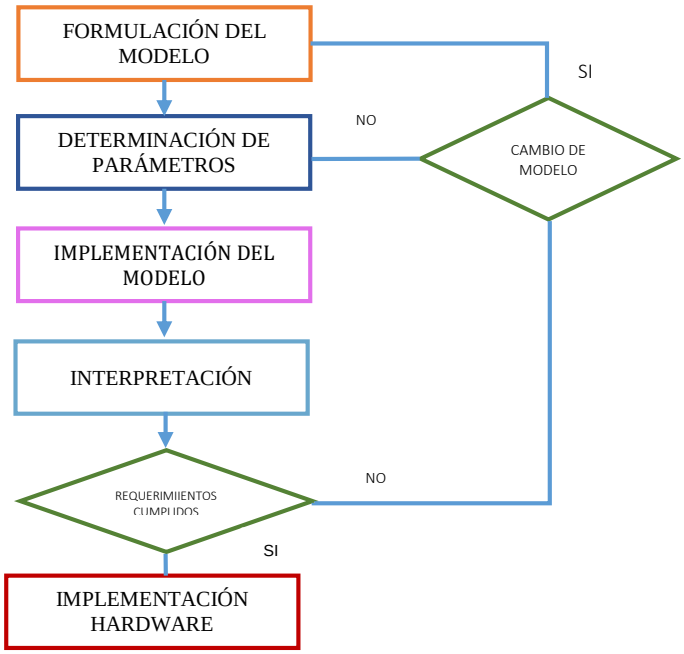


Figura 2.35 Metodología de Simulación.

- **Formulación del Modelo**

Para esta etapa, se plantean dos modelos, con el fin de observar el funcionamiento individual y posteriormente implementarlo a nivel hardware. El primero corresponde al *Encoder BCH* y el segundo, al *Decoder BCH.*

Determinación de Parámetros

Los parámetros que se tienen en cuenta son los expresados en la tabla 2.9, donde la información que se envía al codificador corresponde a 2^k y 2^n al decodificador.

Tabla 2.9 Parámetros para realizar la implementación.

CODIFICADOR/DECODIFICADOR BCH binario						**Fuente de información**
$\boldsymbol{BCH(n,k)}$	**Información enviada**		**Capacidad Correctora t**	**Ciclo de reloj**	**Periodo de bit n**	**Semilla de la fuente**
	2^k	2^n				
$BCH(7,4)$	16	128	1	1	7	Aleatoria
$BCH(15,11)$	2048	32768	1	1	11	Aleatoria
$BCH(15,7)$	128	32768	2	1	7	Aleatoria
$BCH(15,5)$	32	32768	3	1	5	Aleatoria
$BCH(31,26)$	67108864	2147483648	1	1	26	Aleatoria
$BCH(31,21)$	2097152	2147483648	2	1	21	Aleatoria
$BCH(31,16)$	65536	2147483648	3	1	16	Aleatoria

- **Implementación del Modelo**

Al definir los parámetros se procede a implementar el sistema; para esto, se hace uso del bloque *BCH Encoder* que provee la herramienta de Simulink, que compara las salidas de este con el bloque diseñado. La figura 2.36, muestra el sistema para el caso del codificador $\mathrm{BCH}(15,5)$.

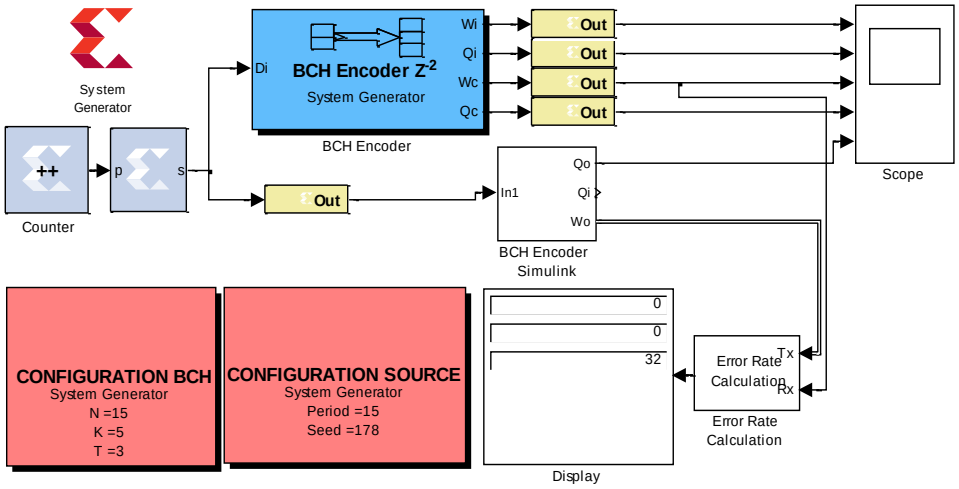


Figura 2.36 Implementación del codificador *BCH encoder.*

Posteriormente, se ilustra el segundo modelo que corresponde al decodificador *BCH* encoder; además se compara la señal de salida del boque de Simulink con el diseñado. En la figura 2.37 se da un ejemplo del decodificador $\mathrm{BCH}(15,5)$.

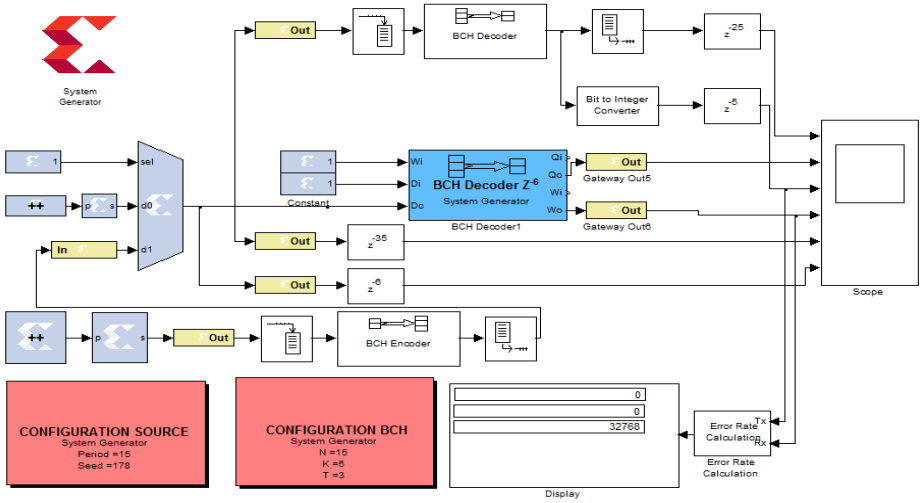


Figura 2.37 Implementación del decodificador *BCH decoder.*

- **Interpretación**

Se comparan los resultados obtenidos por el bloque codificador y decodificador de Simulink frente al diseñado.

Haciendo uso de "Scope", se compararon todos los códigos BCH binarios. Como ejemplo, la figura 2.38 muestra el resultado de un $\mathrm{BCH}(15,5)$ que verifica el correcto funcionamiento del bloque *ENCONDER BCH*. Los dos primeros puntos de medición están situados antes del codificador, el primero, muestra la palabra de información k

transmitida W_i, el segundo, la palabra de información dada en un stream de bits B_i ambas provenientes de la fuente de información. El tercero corresponde a la salida del codificador diseñado W_c que permite observar la palabra codificada n. El cuarto es la palabra dada en un stream de bits. El último es el stream de bits codificados dados por el bloque de Simulink. Se puede observar que la señal proveniente del bloque Simulink y el diseñado son iguales, concluyendo que este funciona correctamente; así mismo se evidencia que la palabra $W_i = 18$, dado en un stream de datos es $B_i = 10010$, el cual se visualiza desde el bit más significativo hasta el menos significativo; es importante mencionar, que al codificarse esa palabra tiene una redundancia de 10 bits, por lo tanto el stream de bits codificados es 1000011101**10010** donde los últimos 5 bits son la palabra de información.

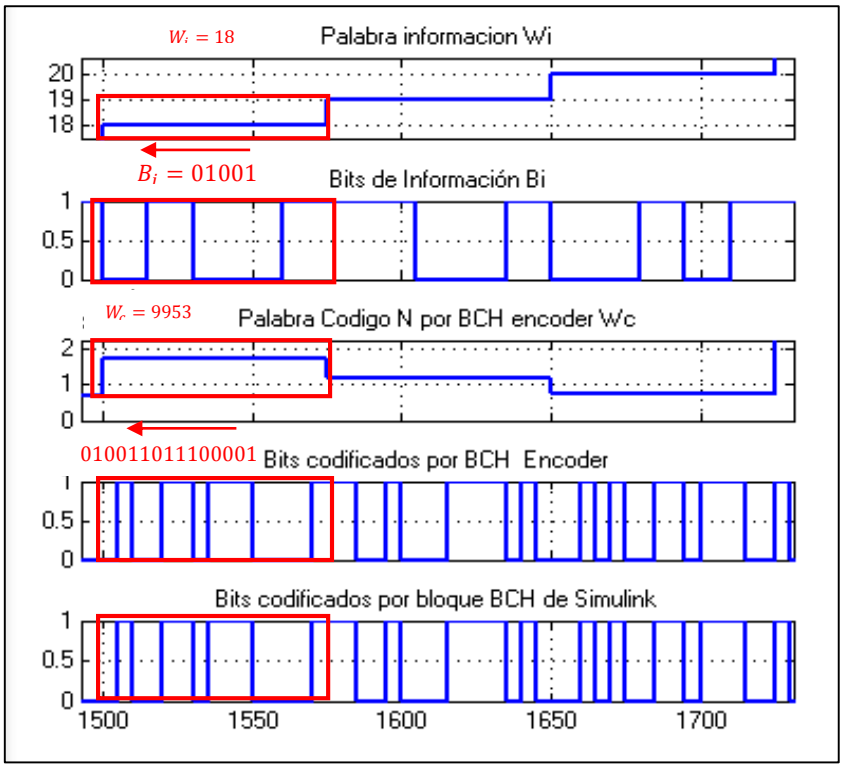


Figura 2.38 Codificación de la información capturas por “Scope”.

La figura 2.39 presenta un ejemplo para el caso de la decodificación. La medición se realiza a la salida del decodificador de Simulink y del bloque diseñado; además se compara la información y se verifica el correcto funcionamiento de este.

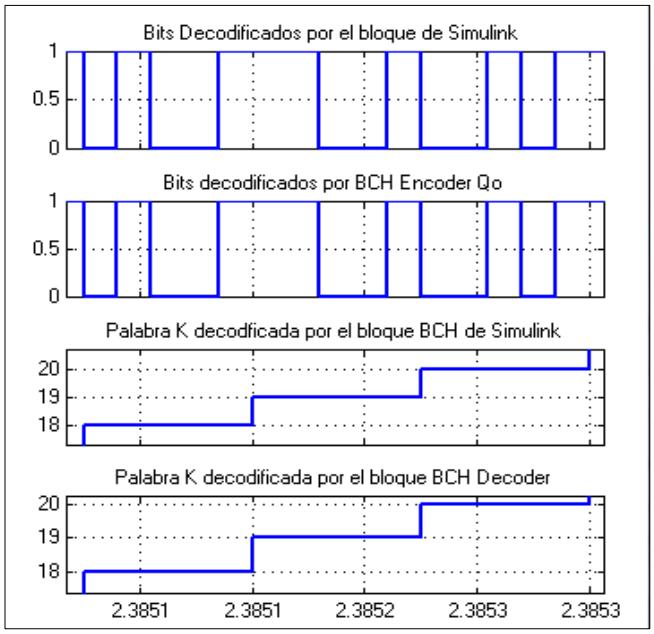


Figura 2.39 Decodificación de la información captura por "Scope".

2.5. IMPLEMENTACIÓN HARDWARE

Para llevar a cabo la implementación de los sistemas de comunicación diseñados sobre el FPGA es necesario generar el archivo de programación bitstream (.bit), el cual contiene el código binario de configuración de la tarjeta y su construcción está compuesta por una serie de procesos, que se describen a continuación.

2.5.1 Fase 4. Implementación física del sistema

La figura 2.40 presenta el proceso para la implementación hardware. Se inicia el procedimiento con la configuración del símbolo *System Generator*, por medio de su panel de opciones, en el cual se establecen los parámetros principales de implementación, como el modo de compilación y la referencia del FPGA, con el fin de obtener un archivo *.xise*, luego éste es leído en *Project Navigator*, que se encarga de la sintetización e implementación del diseño para finalmente realizar la generación del archivo *.bit* y de los *timing constraints* (limitaciones de tiempo), los cuales determinan el retardo máximo que el sistema estaría dispuesto a soportar. Al obtener el archivo *.bit* se procede a cargar el código sobre el FPGA seleccionado (Spartan 3A) por medio de la herramienta iMPACT, que se encarga de realizar el reconocimiento del FPGA y grabar el código. Por último se hace la captura de datos, por medio del *puerto serial* que envía la información del sistema a una aplicación de escritorio *Serial Communication*, encargada de visualizar los datos entregados por el sistema diseñado. Esta sección se encuentra detallada en el apéndice C.

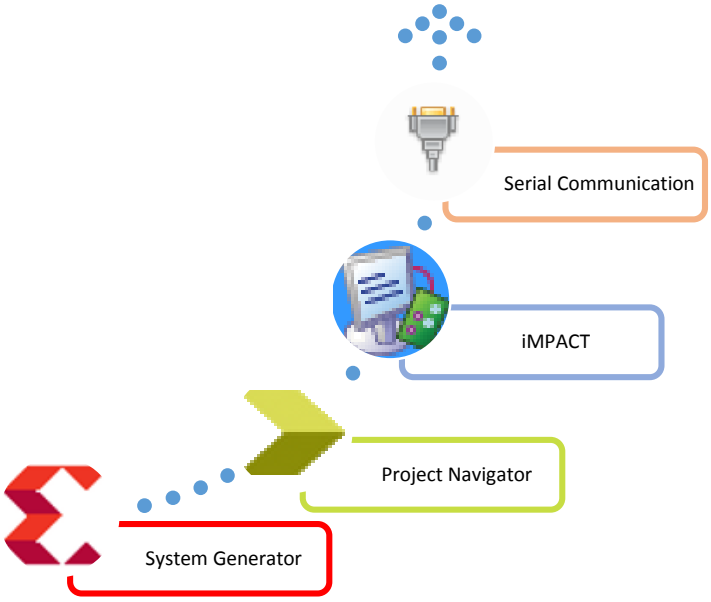


Figura 2.40 Proceso de Implementación Hardware.

2.6. VALIDACIÓN Y PRUEBAS

Una vez implementado el sistema, con el uso de la herramienta de Simulink se realiza la validación del sistema de comunicaciones, en el caso de las pruebas se hace uso de la aplicación *Serial Communication*, la cual permite obtener los valores de BER y WER implementados sobre la FPGA.

2.6.1. Fase 5. Validación

Inicialmente se construyen los modelos en Simulink, de tal forma que al construido en la fase 1, se le agregan: dos bloques *buffer*, el primero agrupa en k bits la información proveniente del generador binario Bernoulli y el segundo une los bits provenientes del demodulador en la palabra de código n; los boques *codificador y decodificador BCH binario*; dos bloques *unbuffer*, donde el primero permite transmitir la palabra código n en bits hacia el modulador FSK/MSK y el segundo la transmisión de la información n en bits hacia el calculador de la tasa de error de bit. En la figura 2.41, se visualiza el modelo para el esquema de modulación FSK y en la figura 2.42, para la modulación MSK.

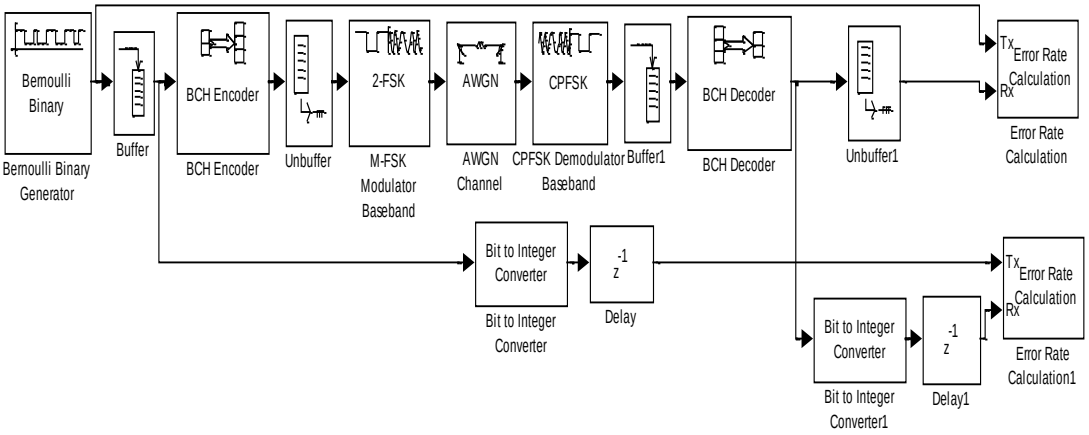


Figura 2.41 Sistema de comunicación banda base con modulación FSK.

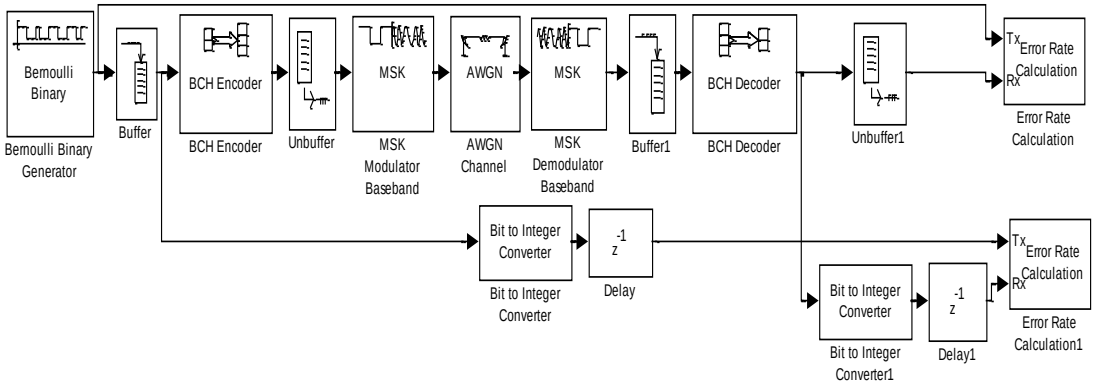


Figura 2.42 Sistema de comunicación banda base con modulación MSK.

Las figuras 2.43, 2.44 y 2.45 muestran los resultados obtenidos para el sistema de comunicación con el código $BCH(7,4)$ y las modulaciones FSK con índices de modulación $h = 0.5$ y $h = 0.25$ y MSK comparando las curvas de desempeño a nivel de la BER del modelo de Simulink con el implementado en el FPGA; se tiene como resultado que el sistema implementado sigue el mismo comportamiento que el de Simulink.

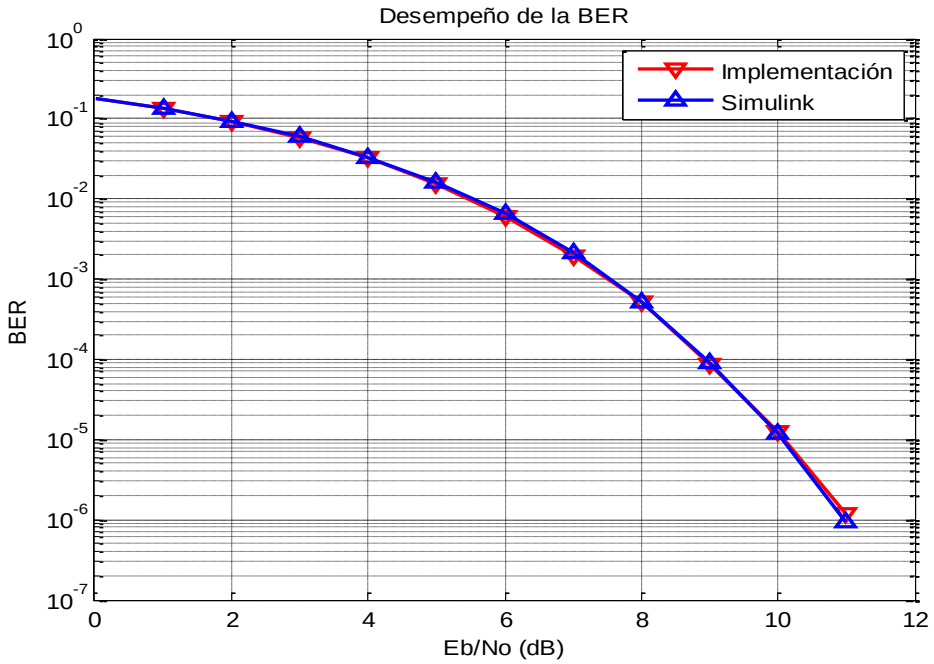


Figura 2.43 Curvas de desempeño de la BER del código $BCH(7,4)$ con FSK $h = 0.5$, Simulink e Implementada.

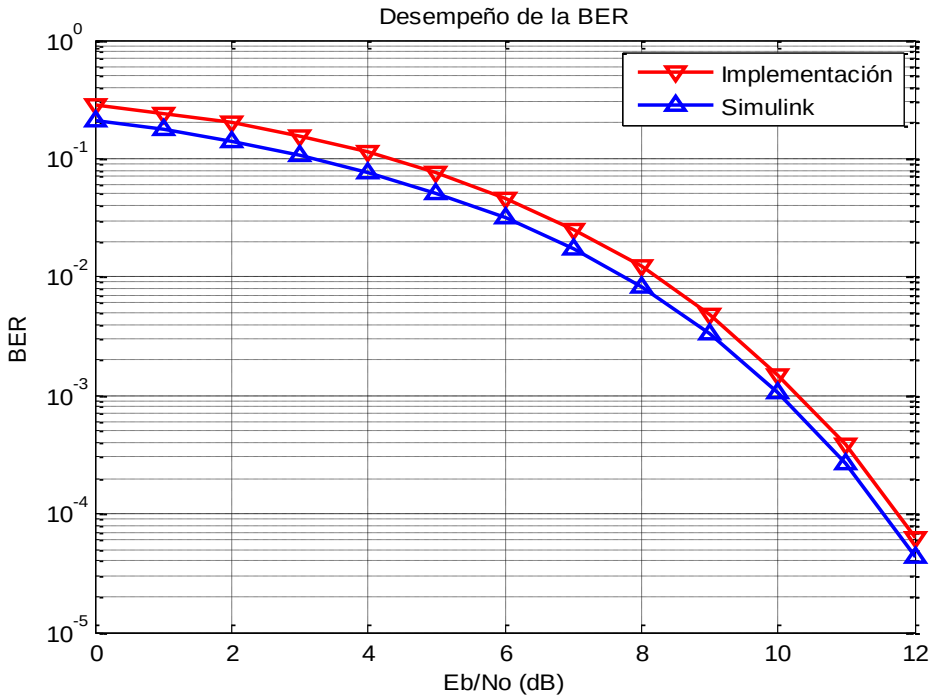


Figura 2.44 Curva de desempeño de la BER del código $BCH(7,4)$ con FSK $h = 0.25$, Simulink e Implementada.

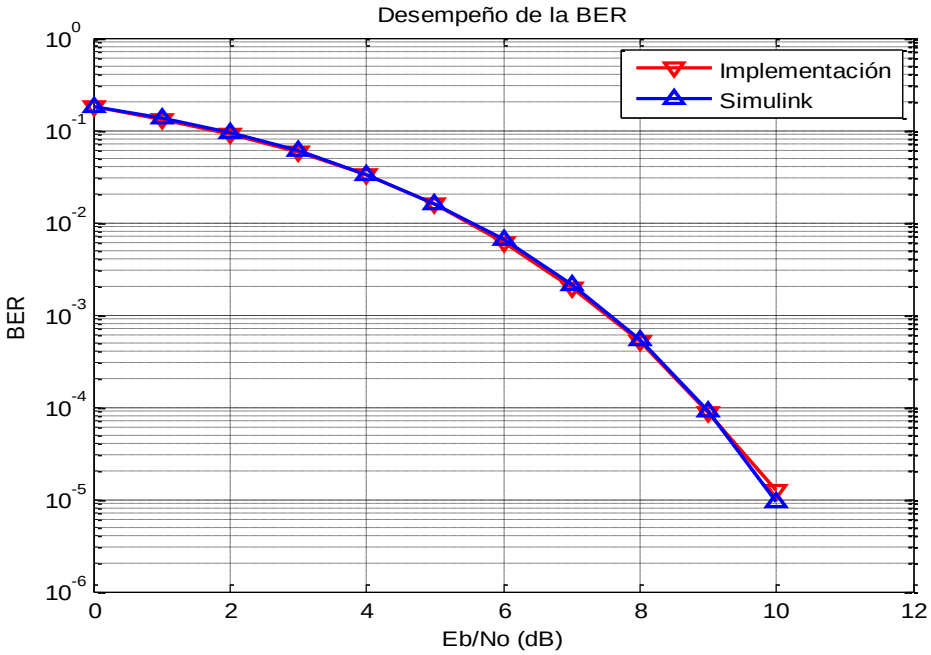


Figura 2.45 Curvas de desempeño a nivel de la BER del código $BCH(7,4)$ con MSK de Simulink e Implementada.

Las figuras 2.46, 2.47 y 2.48 dan a conocer los resultados obtenidos para el sistema de comunicación con el código $\mathrm{BCH}(7,4)$ y las modulaciones FSK con $h = 0.5$ y $h = 0.25$ y MSK, comparando las curvas de desempeño a nivel de la WER del modelo de Simulink con el implementado en el FPGA. El desempeño del sistema implementado sigue el mismo comportamiento que el de Simulink.

Los valores de BER y WER obtenidos se describen en la sección D.1 del apéndice D.

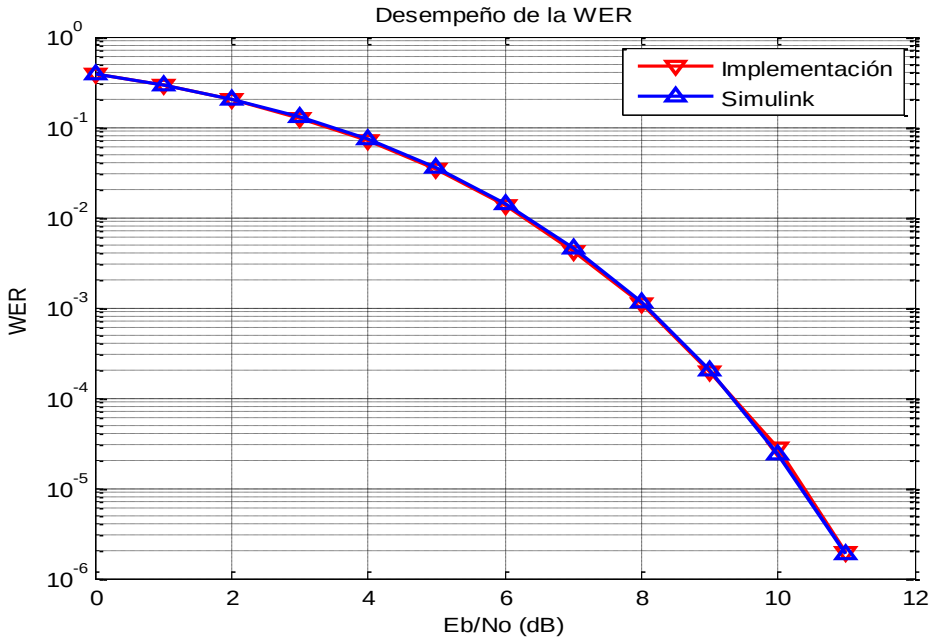


Figura 2. 46 Curvas de desempeño de la WER del código $BCH(7,4)$ con FSK $h = 0.5$, Simulink e Implementada.

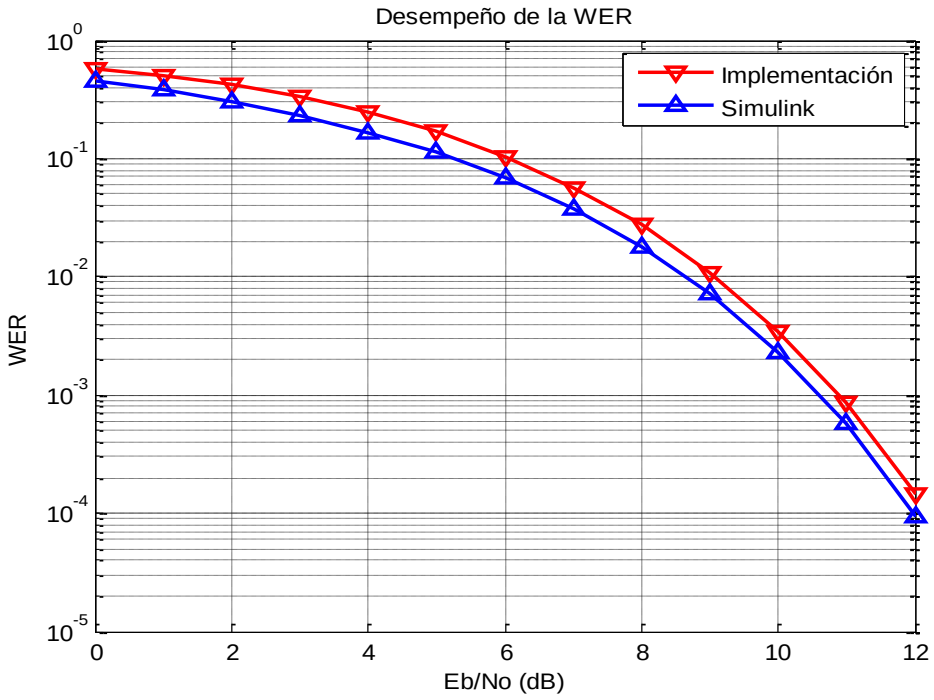


Figura 2.47 Curva de desempeño de la WER del código $BCH(7,4)$ con FSK $h = 0.25$, Simulink e Implementada.

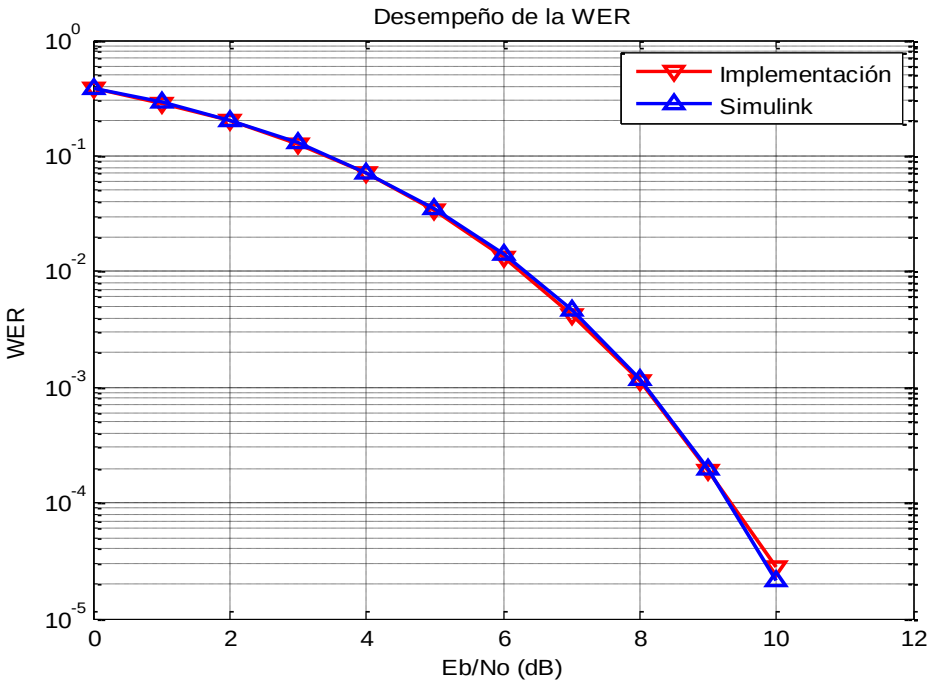
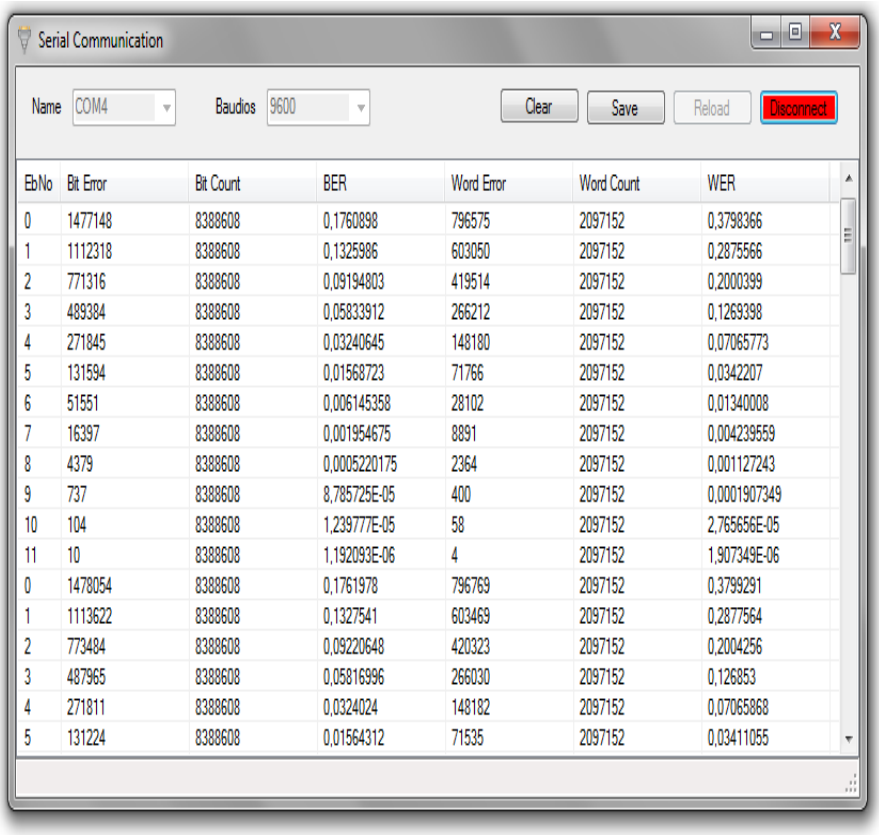


Figura 2.48 Curvas de desempeño a nivel de la WER del código $BCH(7,4)$ con MSK, Simulink e Implementada.

2.6.2. Fase 6. Pruebas

Para realizar las pruebas del sistema de comunicaciones banda base con codificación BCH binaria se utiliza la aplicación *Serial Communication.* En la figura 2.49, se puede observar la captura de los datos obtenidos del FPGA con el código $BCH(7,4)$ y modulación MSK. De esta forma, se visualizan los datos correspondientes al valor del E_b/N_0, bits y palabras erradas, bits y palabras transmitidas y el respectivo valor de BER y WER.



EbNo	Bit Error	Bit Count	BER	Word Error	Word Count	WER
0	1477148	8388608	0,1760898	796575	2097152	0,3798366
1	1112318	8388608	0,1325986	603050	2097152	0,2875566
2	771316	8388608	0,09194803	419514	2097152	0,2000399
3	489384	8388608	0,05833912	266212	2097152	0,1269398
4	271845	8388608	0,03240645	148180	2097152	0,07065773
5	131594	8388608	0,01568723	71766	2097152	0,0342207
6	51551	8388608	0,006145358	28102	2097152	0,01340008
7	16397	8388608	0,001954675	8891	2097152	0,004239559
8	4379	8388608	0,0005220175	2364	2097152	0,001127243
9	737	8388608	8,785725E-05	400	2097152	0,0001907349
10	104	8388608	1,239777E-05	58	2097152	2,765656E-05
11	10	8388608	1,192093E-06	4	2097152	1,907349E-06
0	1478054	8388608	0,1761978	796769	2097152	0,3799291
1	1113622	8388608	0,1327541	603469	2097152	0,2877564
2	773484	8388608	0,09220648	420323	2097152	0,2004256
3	487965	8388608	0,05816996	266030	2097152	0,126853
4	271811	8388608	0,0324024	148182	2097152	0,07065868
5	131224	8388608	0,01564312	71535	2097152	0,03411055

Figura 2.49 Funcionamiento de la aplicación *Serial Communication*.

Una vez finalizada la fase 6, se continúa con el análisis de los resultados obtenidos al implementar el sistema de comunicaciones banda base con codificación BCH binaria, el cual se desarrolla en el siguiente capítulo.

Capítulo 3.

EXPERIMENTACIÓN Y ANÁLISIS DE RESULTADOS

3.1. PLAN DE PRUEBAS

Al validarse el sistema básico de comunicaciones banda base de la codificación BCH binaria con los esquemas de modulación MSK/FSK en simulación e implementación sobre el FPGA, se definen los escenarios de pruebas, variando dos parámetros: capacidad correctora del código t y tipo de modulación.

3.1.1. Definición de Escenarios

En la tabla 3.1 se definen los escenarios desarrollados para los diferentes códigos BCH binarios implementados con las respectivas modulaciones desarrolladas en la fase 1. De esta manera; en el primero se hace una comparación del desempeño a nivel de BER y WER del sistema implementado frente a las curvas teóricas; el segundo, presenta la comparación de las curvas de desempeño obtenidas de la herramienta Simulink, System Generator y del sistema implementado en el FPGA con los códigos BCH binarios construidos y la modulación FSK, $h = 0.5$; en el tercero a diferencia del anterior, se realizan las comparaciones con $h = 0.25$; el cuarto, compara el desempeño del $\mathrm{BCH}(15, k)$ con MSK/FSK $h = 0.5$ y en el último se compara la codificación $\mathrm{BCH}(31, 26)$, $\mathrm{BCH}(31, 21)$, $\mathrm{BCH}(31, 16)$ con MSK, para analizar el desempeño según la capacidad correctora de los códigos implementados.

Tabla 3.1 Escenarios de pruebas.

ESCENARIOS	MODULACIÓN			CODIFICACIÓN						
	FSK		MSK	BCH BINARIO						
				t=1			t=2		t=3	
	h=0.5	h=0.25		(7,4)	(15,11)	(31,26)	(15,7)	(31,21)	(15,5)	(31,16)
1	x		x	x						
2	x			x	x	x	x	x	x	x
3		x		x	x	x	x	x	x	x
4	x		x		x		x		x	
5			x			x		x		x

3.2. RESULTADOS Y ANÁLISIS

En esta sección se comparan los resultados obtenidos al implementar el sistema en Simulink, System Generator y en el FPGA; en cada uno de los casos se ejecutaron 20 simulaciones y se tuvo en cuenta la media como dato de tabulación. Para los cinco escenarios planteados la información transmitida corresponde a 8'388.608 (2^{23}) bits, variando las semillas del generador de bits y del canal AWGN.

3.2.1. Escenario 1

Inicialmente se obtiene la curva teórica del sistema mediante la probabilidad de error de bit para una codificación BCH binaria [28], que resulta de la ecuación 3.1:

$$\frac{1}{k}P_w \leq P_b \leq P_w, \tag{3.1}$$

donde, la probabilidad de error en una palabra código es:

$$P_w \leq \sum_{i=t+1}^{n} \binom{n}{i} p^i (1-p)^{n-i}, \tag{3.2}$$

de esta manera la detección de error en un bit de información está dada por la ecuación 3.1, de la cual se toma la cota superior, debido a que la inferior es utilizada para codificación Gray, como se indica en la ecuación 3.3.

$$P_b \leq \sum_{i=t+1}^{n} \binom{n}{i} p^i (1-p)^{n-i}, \tag{3.3}$$

Siendo:

$i =$ Número de bits errados en una palabra código de n bits.
$n =$ Longitud del código.
$t =$ Número de errores máximos a corregir.
$p =$ Probabilidad de error en un bit de la modulación utilizada.

Donde la probabilidad de error de bit para un sistema FSK banda base con fase continúa y detección coherente está dada por:

$$p \approx Q\left(\sqrt{d_{mín}\frac{E_b}{N_o}}\right) \tag{3.4}$$

Siendo:

$d_{mín}$: Distancia mínima entre símbolos.
E_b: Energía de bit.
N_o: Potencia de ruido.

Para el caso de la señal FSK de fase continua binaria la mínima distancia euclidiana está dada por:

$$d_{mín} = 2\big(1 - sinc(2h)\big).$$

Dado que la energía de bit se ve afectada por la tasa de codificación $nE_b' = kE_b$, entonces $E_b' = \frac{k}{n}E_b$.

Ahora la probabilidad de error en un bit en el sistema con codificación lineal es:

$$p \approx Q\left(\sqrt{d_{mín}\frac{E_b'}{N_o}}\right) \approx Q\left(\sqrt{d_{mín}\frac{k}{n}\frac{E_b}{N_o}}\right). \tag{3.5}$$

Finalmente la probabilidad de error de bit para un sistema de MSK con detección coherente está dada por:

$$p = erfc\left(\sqrt{\frac{E_b}{N_o}}\right)\left[1 - \frac{1}{2}erfc\left(\sqrt{\frac{E_b}{N_o}}\right)\right], \tag{3.6}$$

Entonces la probabilidad de error en un bit en el sistema con codificación lineal es:

$$p = erfc\left(\sqrt{\frac{E_b'}{N_o}}\right)\left[1 - \frac{1}{2}erfc\left(\sqrt{\frac{E_b'}{N_o}}\right)\right] = erfc\left(\sqrt{\frac{k}{n}\frac{E_b}{N_o}}\right)\left[1 - \frac{1}{2}erfc\left(\sqrt{\frac{k}{n}\frac{E_b}{N_o}}\right)\right]. \tag{3.7}$$

La probabilidad de error de bit está directamente relacionada con la probabilidad de error en una palabra código dada por un factor de escalonamiento [5] así:

$$BER = \frac{2^k/2}{2^k - 1}WER. \tag{3.8}$$

En la figura 3.1 se expone un ejemplo para el código $\mathrm{BCH}(7,4)$ de la relación existente entre la BER y la WER afectadas por un factor de escalonamiento aproximadamente igual a 1.875.

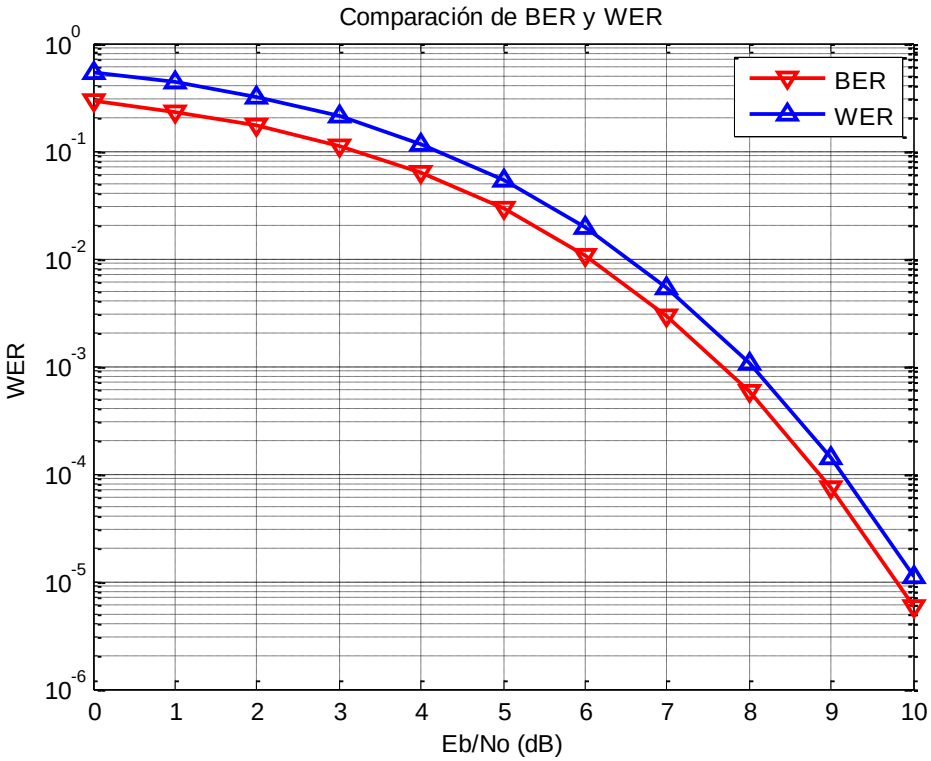


Figura 3.1 Curva comparativa de la BER y WER para $BCH(7,4)$ con MSK.

Una vez obtenida la curva teórica, las figuras 3.2 y 3.3 presentan la validación de las curvas de desempeño a nivel de la BER y WER en función de E_b/N_o de la codificación $\mathrm{BCH}(7,4)$ con esquema de modulación MSK obtenidas en Simulink, System Generator y el FPGA, las cuales son comparadas con la curva teórica aproximada. En estas se observa que las tres curvas presentan un comportamiento similar a la teórica comprobándose que el sistema implementado es correcto.

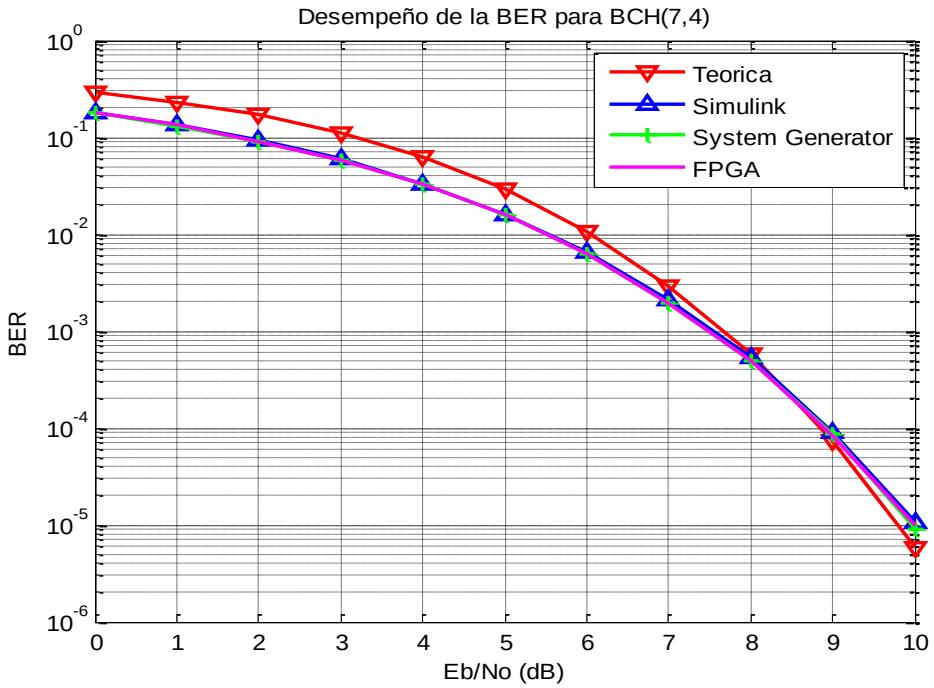


Figura 3.2 Curvas de desempeño de la BER del código $BCH(7,4)$ con MSK, teórica, Simulink, System Generator y FPGA.

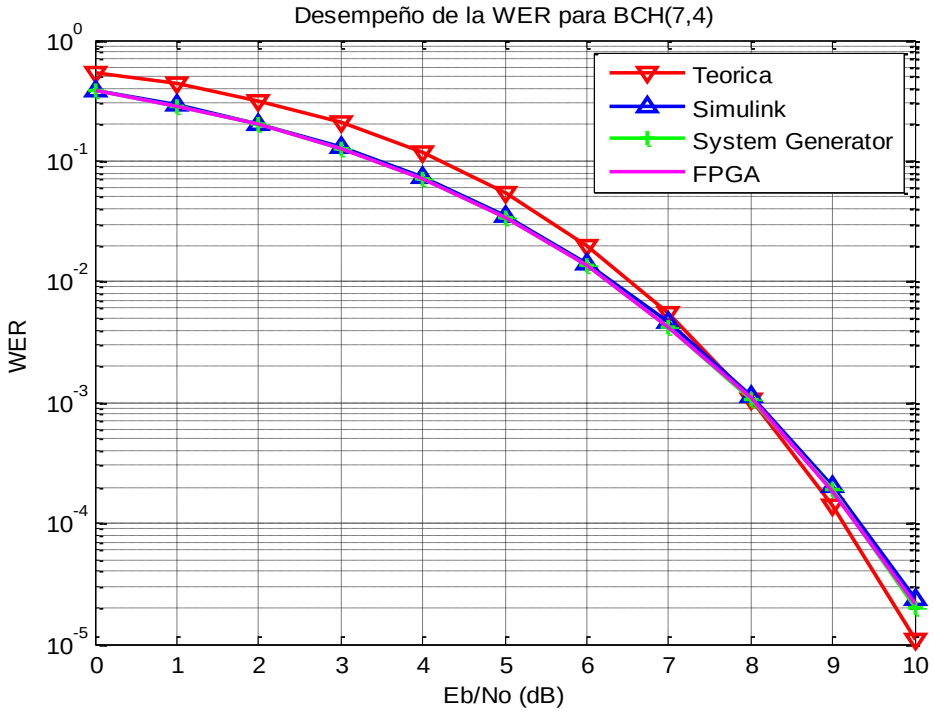


Figura 3.3 Curvas de desempeño de la WER del código $BCH(7,4)$ con MSK teórica, Simulink, System Generator y FPGA.

Las figuras 3.4 y 3.5 ilustran la validación de las curvas de desempeño a nivel de la BER y WER en función de (E_b/N_o) de la codificación $\mathrm{BCH}(7,4)$ con el esquema de modulación FSK, $h = 0.5$ obtenidas en Simulink, System Generator y el FPGA, las cuales son comparadas con la teórica.

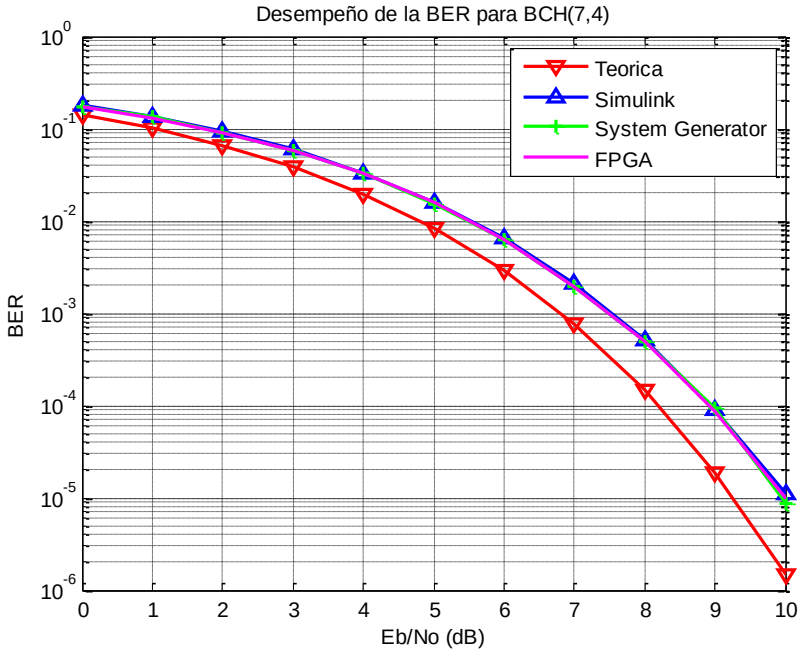


Figura 3.4 Curva de desempeño de la BER del código $BCH(7,4)$ con FSK, $h = 0.5$, teórica, Simulink, System Generator y FPGA.

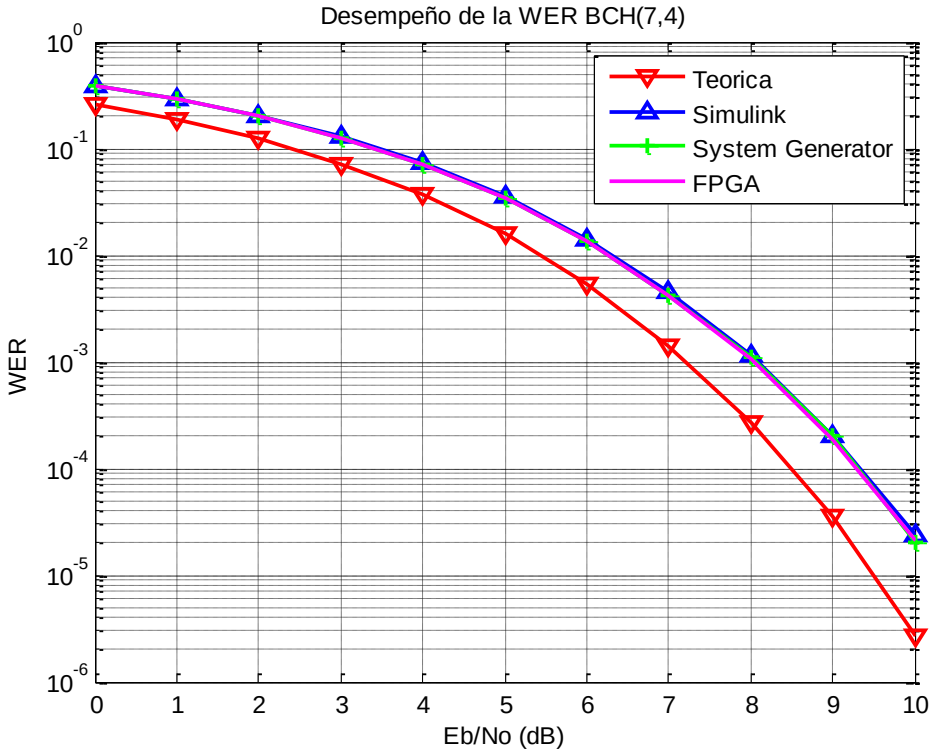


Figura 3.5 Curva de desempeño de la WER del código $BCH(7,4)$ con FSK, $h = 0.5$, teórica, Simulink, System Generator y FPGA.

Es de resaltar que la diferencia entre las tres curvas simuladas (Simulink, System Generator, FPGA) respecto a la teórica, se debe a tres factores: primero, la probabilidad de error de la modulación FSK de la fase 1 es una aproximación; segundo el algoritmo de demodulación implementado incide en la etapa de decodificación y finalmente se trabaja con un canal AWGN, el cual no permite un control de errores debido a que adiciona errores de forma aleatoria.

Para los siguientes escenarios es más relevante la comparación de la curva implementada de la fase 1 del proyecto "Análisis del Desempeño de un Sistema de Comunicaciones con Modulación MSK/FSK basado en Hardware Reconfigurable" para un sistema sin codificar, frente a las curvas alcanzadas en el actual trabajo de grado, para un sistema codificado, correspondientes a Simulink, System Generator y FPGA a nivel de BER. La WER no se tiene en cuenta por dos razones: primero, matemáticamente se demostró la relación existente con la BER por medio del factor de escalonamiento y segundo, el parámetro de desempeño del sistema sin codificar corresponde a la BER.

3.2.2. Escenario 2

Se muestran las curvas de desempeño para los códigos $\mathrm{BCH}(7,4)$, $\mathrm{BCH}(15,k)$, $\mathrm{BCH}(31,26)$, $\mathrm{BCH}(31,21)$ y $\mathrm{BCH}(31,16)$ con la modulación FSK, $h = 0.5$, obtenidas en las herramientas Simulink, System Generator y el sistema implementado en el FPGA; las cuales se comparan con la curva de desempeño sin codificar obtenida en la fase 1.

Se evidencia que las curvas de desempeño de los códigos $\mathrm{BCH}(7,4)$, $\mathrm{BCH}(15,11)$ y $\mathrm{BCH}(31,26)$ correspondientes a las figuras 3.6, 3.7 y 3.8 respectivamente, realizadas en Simulink, System Generator y FPGA coinciden, concluyendo que el sistema simulado e implementado funciona de forma correcta. Además, se puede observar que las curvas de desempeño no presentan ninguna mejora respecto a la que no está codificada; lo cual se da principalmente porque el canal AWGN introduce más de un error y la capacidad correctora que presentan estos códigos $(t = 1)$ es muy limitada para un sistema sin control de errores.

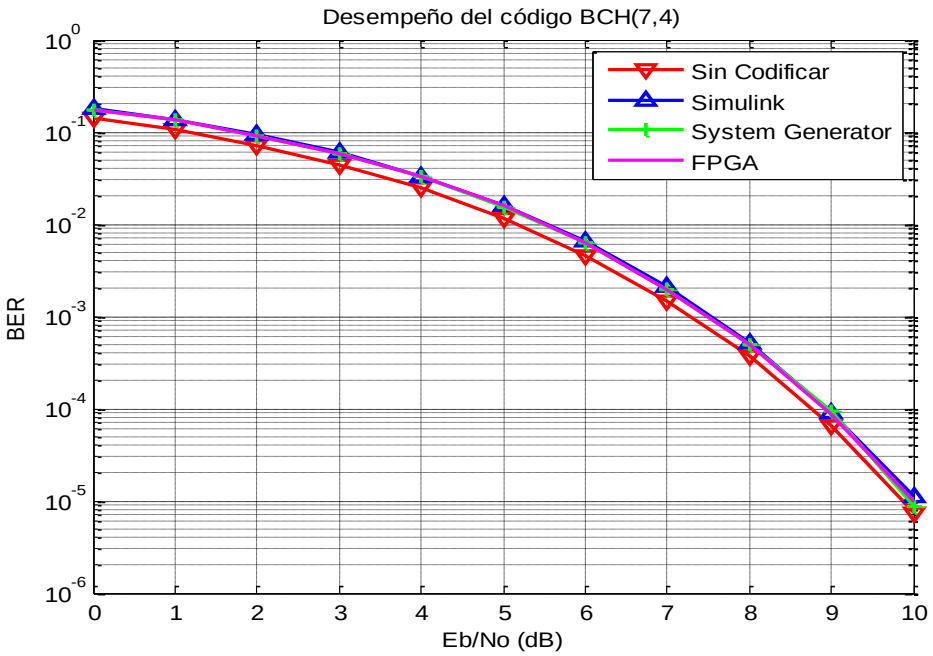


Figura 3.6 Desempeño de la BER del código $BCH(7,4)$ con modulación FSK, $h = 0.5$, Simulink, System Generator y FPGA.

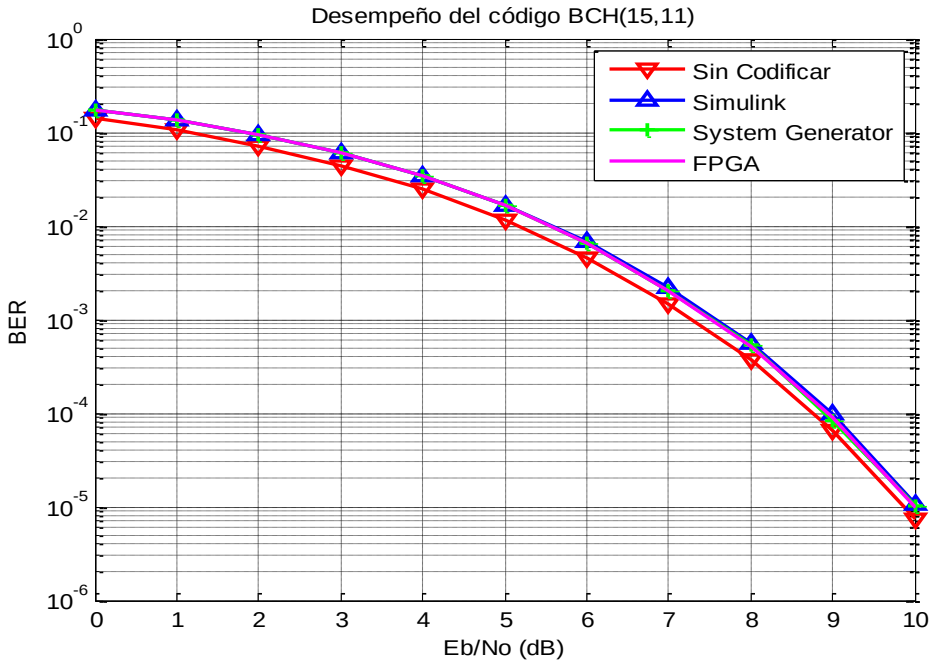


Figura 3.7. Desempeño de la BER del código $BCH(15,11)$ con modulación FSK, $h = 0.5$ Simulink, System Generator y FPGA.

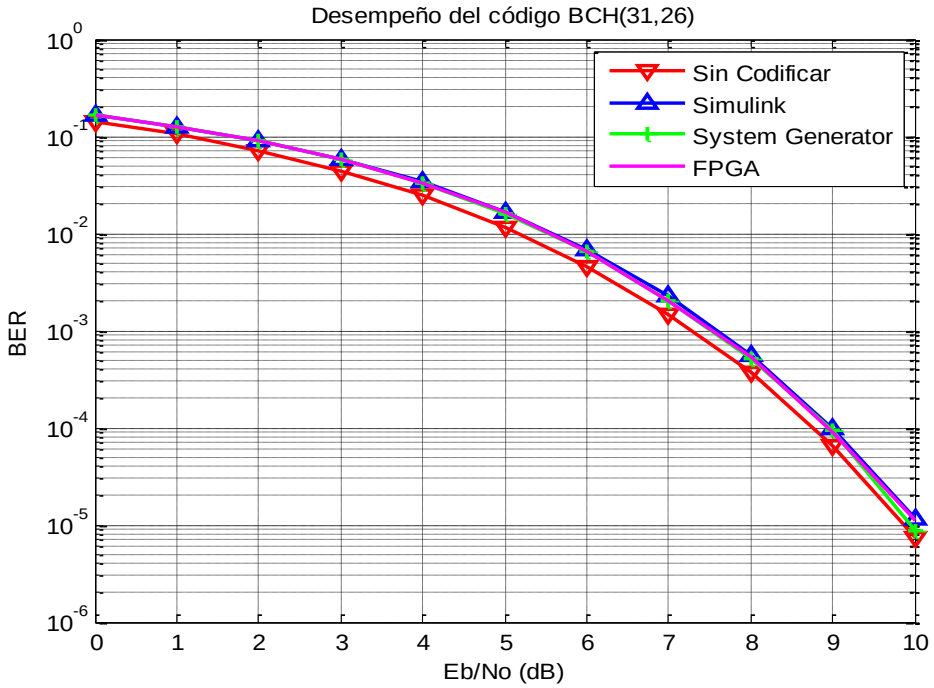


Figura 3.8 Desempeño de la BER del código $BCH(31,26)$ con modulación FSK, $h = 0.5$, Simulink, System Generator y FPGA.

Las curvas de desempeño de los códigos $BCH(15,5), BCH(15,7), BCH(31,21)$ y $\mathrm{BCH}(31,16)$ correspondientes a las figuras 3.9, 3.10, 3.11 y 3.12 respectivamente, realizadas en Simulink, System Generator y FPGA, presentan una leve diferencia de 0.3 dB respecto a la curva de Simulink, debido a que las técnicas de demodulación y de decodificación utilizadas son diferentes. Para el caso de la decodificación en Simulink se usa el algoritmo Berlekamp para encontrar el polinomio localizador de error, mientras que el utilizado en System Generator es Berlekamp-Massey.

Por otro lado se observa una mejora significativa respecto a la curva de desempeño sin codificar dado que estos códigos tienen una mayor capacidad correctora, por lo cual la ganancia de codificación, se incrementa a medida que la relación E_b/N_o aumenta.

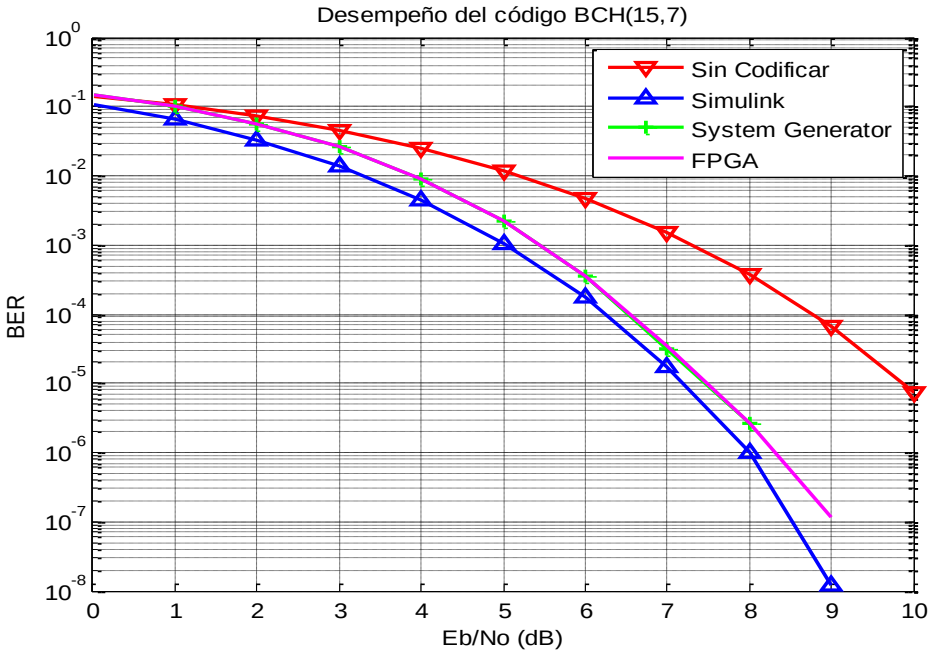


Figura 3.9 Desempeño de la BER del código $BCH(15,7)$ con modulación FSK, $h = 0.5$, Simulink, System Generator y FPGA.

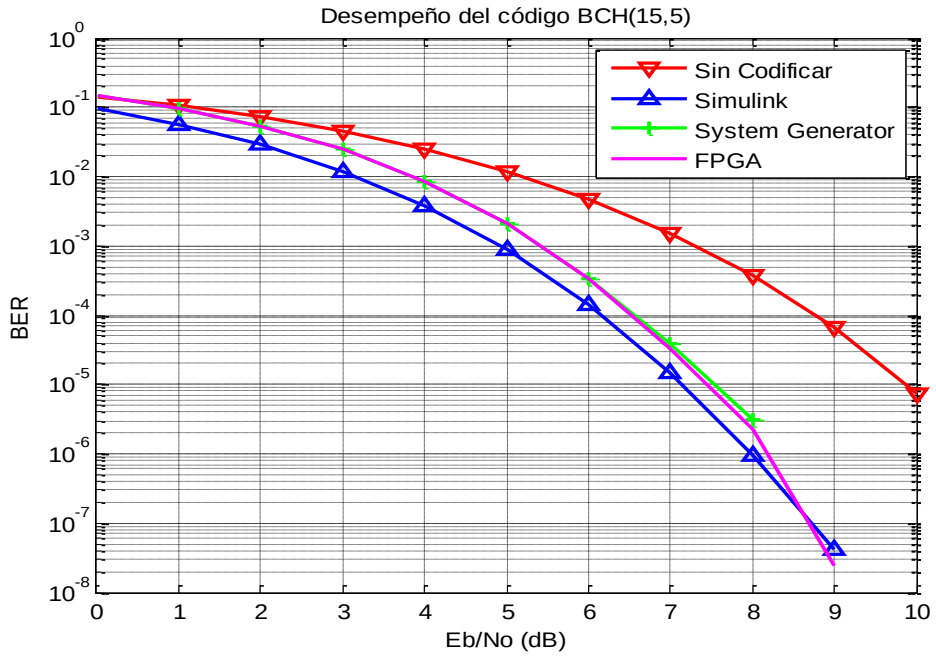


Figura 3.10 Desempeño de la BER del código $BCH(15,5)$ con modulación FSK, $h = 0.5$ Simulink, System Generator y FPGA.

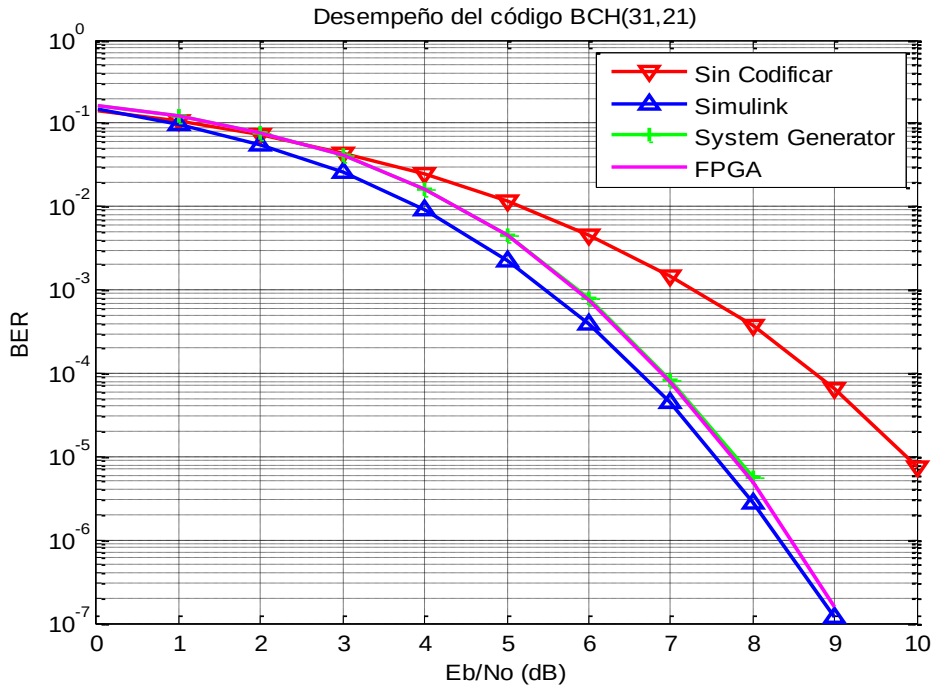


Figura 3.11 Desempeño de la BER del código $BCH(31,21)$ con modulación FSK, $h = 0.5$, Simulink, System Generator y FPGA.

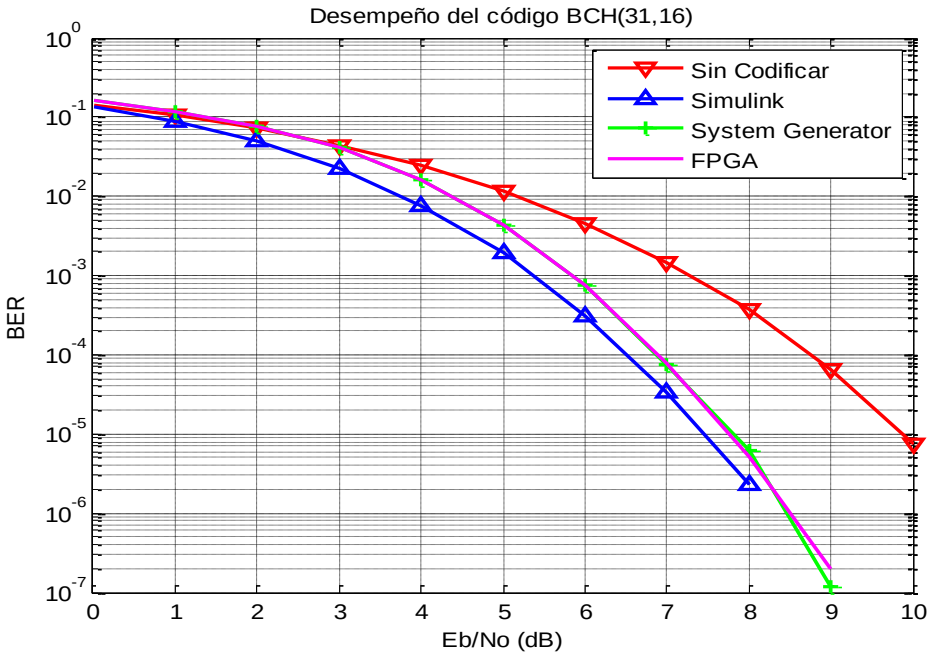


Figura 3.12 Desempeño de la BER del código $BCH(31,16)$ con modulación FSK, $h = 0.5$, Simulink, System Generator y FPGA.

3.2.3. Escenario 3

Para este escenario se ilustran las curvas de desempeño obtenidas de los códigos $\mathrm{BCH}(7,4)$, $\mathrm{BCH}(15,k)$, $\mathrm{BCH}\ (31,26)$, $\mathrm{BCH}\ (31,21)$ y $\mathrm{BCH}\ (31,16)$ con la modulación FSK, $h = 0.25$, sobre las herramientas Simulink, System Generator y el sistema implementado en el FPGA; las cuales se comparan con la curva de desempeño sin codificar obtenida en la fase 1.

Las curvas de desempeño de System Generator y FPGA de las figuras 3.13, 3.14 y 3.15 no presentan mejora frente a la curva sin codificar, sin embargo las de Simulink presentan una mejora aproximada de 0.3 dB, debido a que en la fase 1 se evidenció que ésta última con modulación FSK, $h = 0.25$ presenta un mejor desempeño que la implementada.

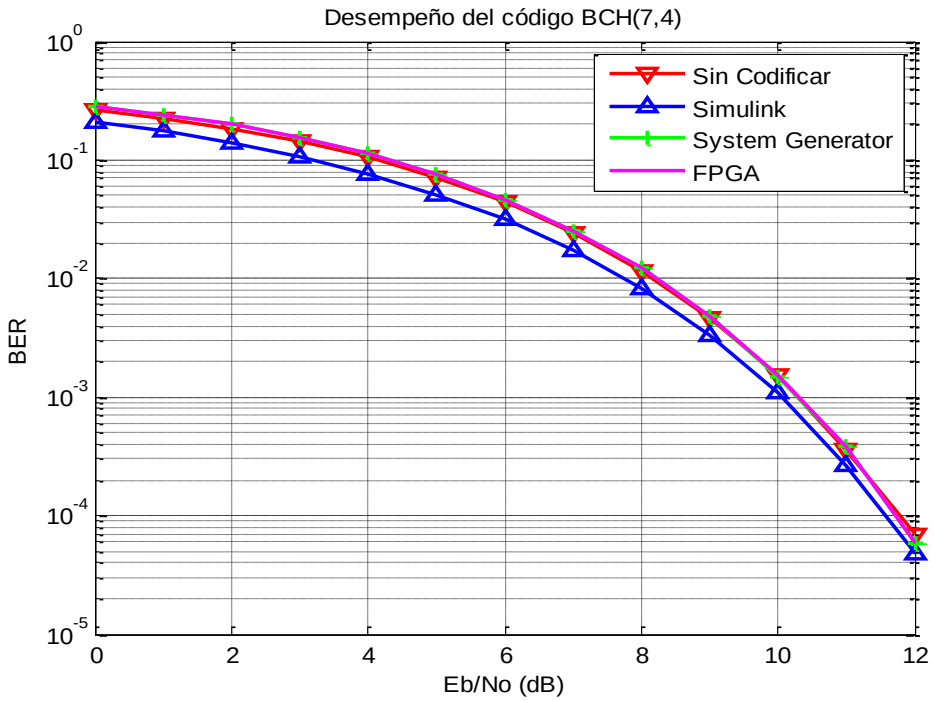


Figura 3.13 Desempeño de la BER del código $BCH(7,4)$ con modulación FSK, $h = 0.25$, Simulink, System Generator y FPGA.

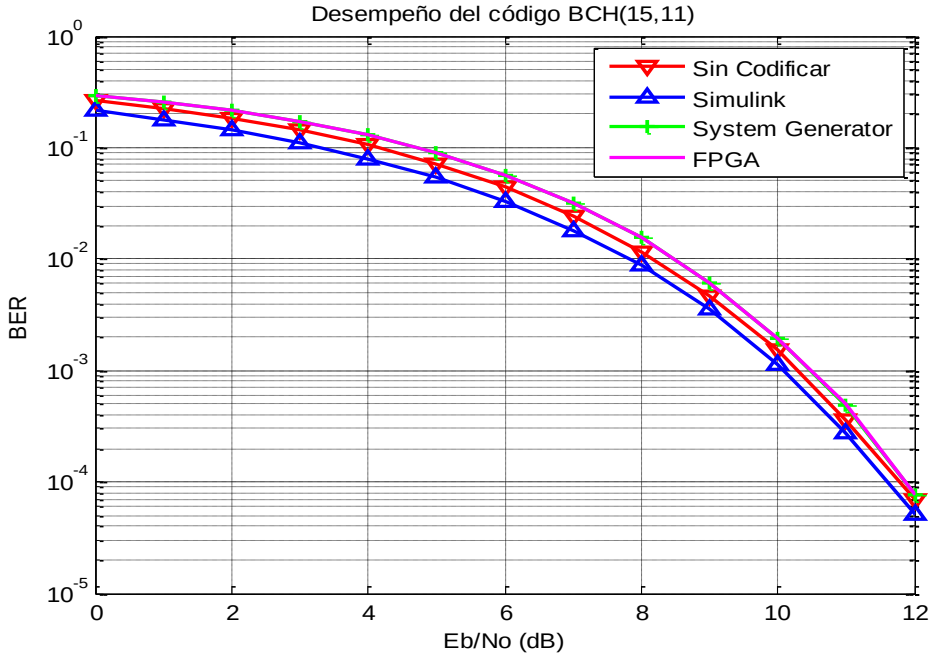


Figura 3.14 Desempeño de la BER del código $BCH(15,11)$ con modulación FSK, $h = 0.25$, Simulink, System Generator y FPGA.

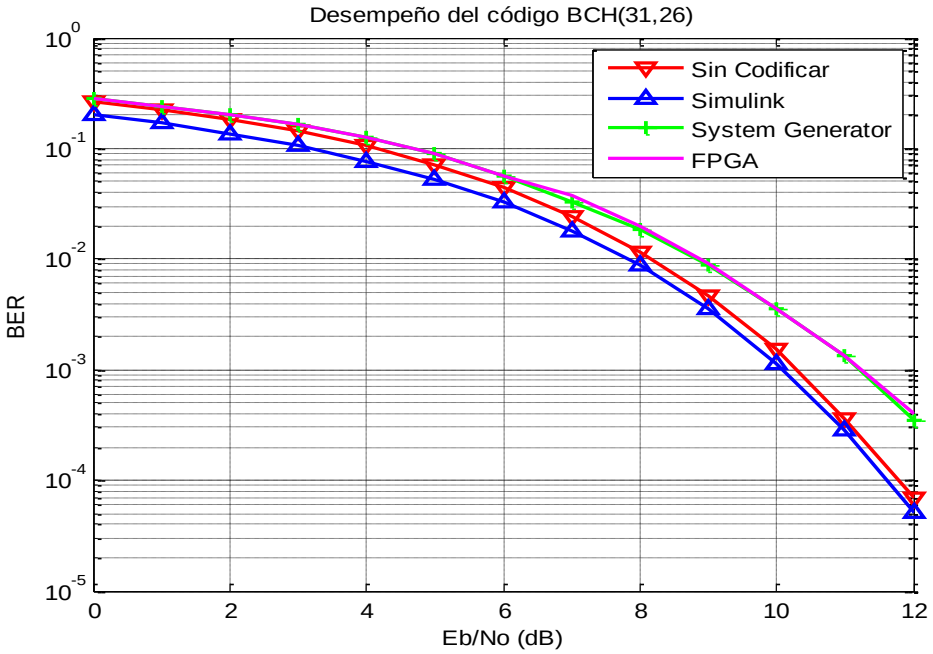


Figura 3.15 Desempeño de la BER del código $BCH(31,26)$ con modulación FSK, $h = 0.25$, Simulink, System Generator y FPGA.

Al igual que en el escenario anterior las curvas de desempeño de las figuras 3.16, 3.17, 3.18 y 3.19 presentan una mejora frente a la curva sin codificar para una capacidad correctora diferente de uno, observando que la curva de Simulink presenta un mejor desempeño.

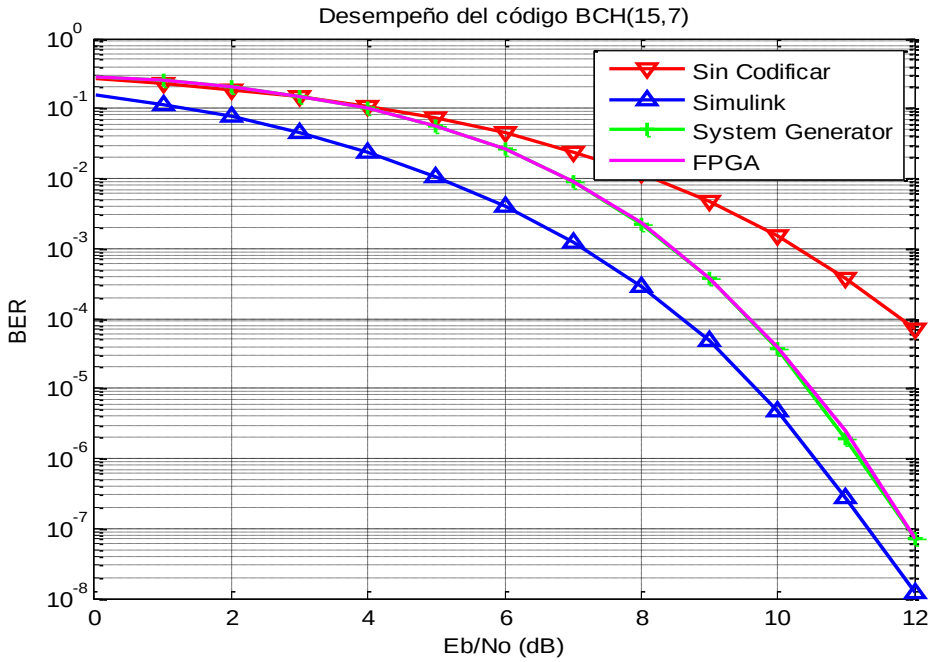


Figura 3.16 Desempeño de la BER del código $BCH(15,7)$ con modulación FSK, $\mathrm{h} = 0.25$, Simulink, System Generator y FPGA.

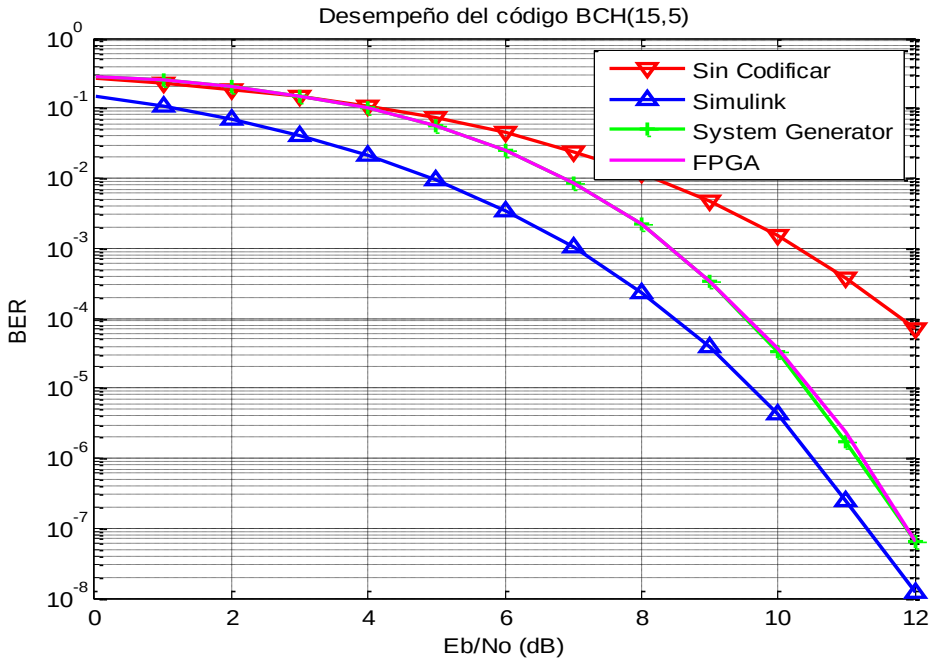


Figura 3.17 Desempeño de la BER del código $BCH(15,5)$ con modulación FSK, $h = 0.25$, Simulink, System Generator y FPGA.

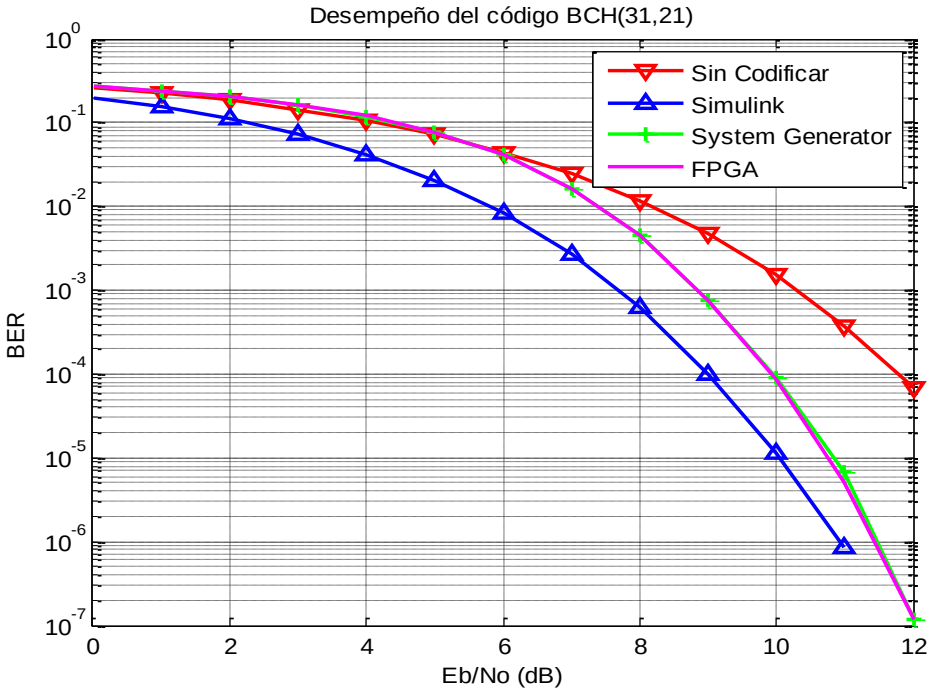


Figura 3.18 Desempeño a nivel de la BER del código $BCH(31,21)$ con modulación FSK, $h = 0.25$, Simulink, System Generator y FPGA.

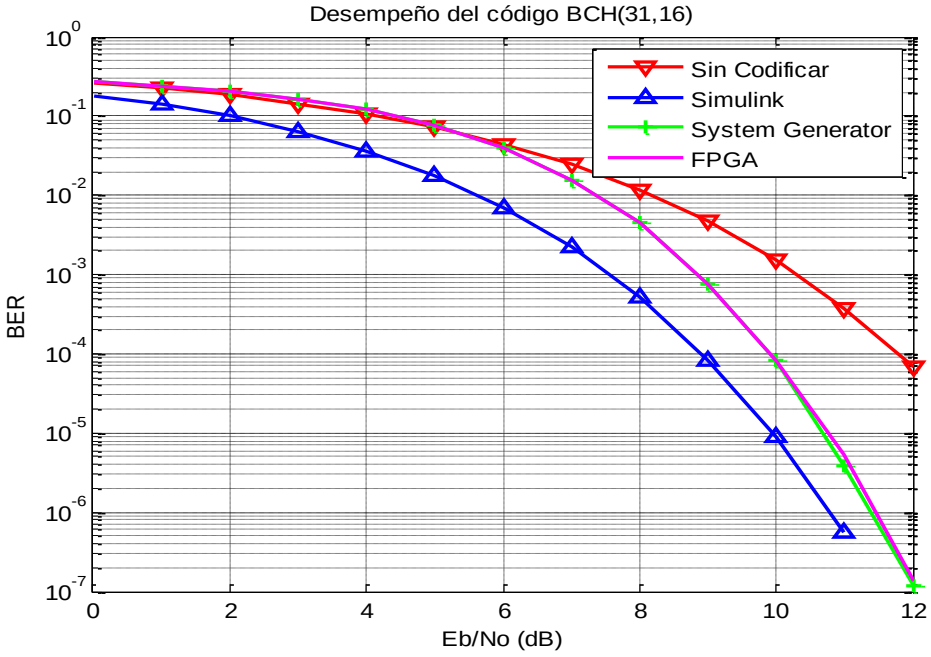


Figura 3.19 Desempeño a nivel de la BER del código $BCH(31,16)$ con modulación FSK, $h = 0.25$, Simulink, System Generator y FPGA.

3.2.4. Escenario 4

En este escenario se puede apreciar que para los códigos BCH binarios las curvas de desempeño de MSK y FSK presentan el mismo desempeño debido a que MSK es un caso particular de FSK con $h = 0.5$. En las figuras 3.20, 3.21 y 3.22 se exponen este comportamiento para el código $\mathrm{BCH}(15, k)$.

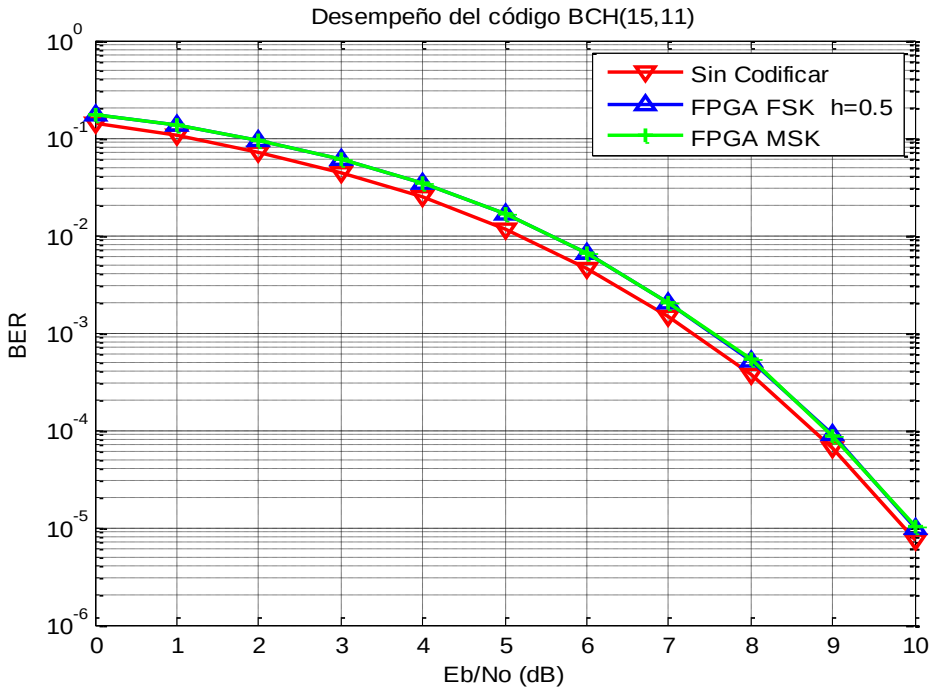


Figura 3.20 Comparación de la curva de desempeño del código $BCH(15,11)$ con modulación MSK y FSK, $h = 0.5$, Simulink, System Generator y FPGA.

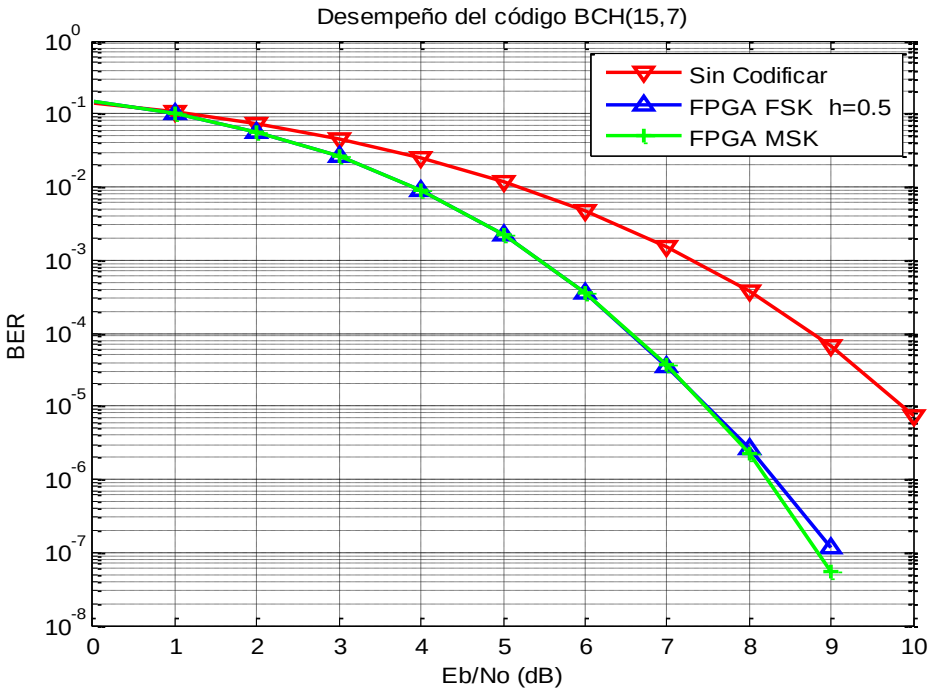


Figura 3.21 Comparación de la curva de desempeño del código $BCH(15,7)$ con modulación MSK y FSK, $h = 0.5$, Simulink, System Generator y FPGA.

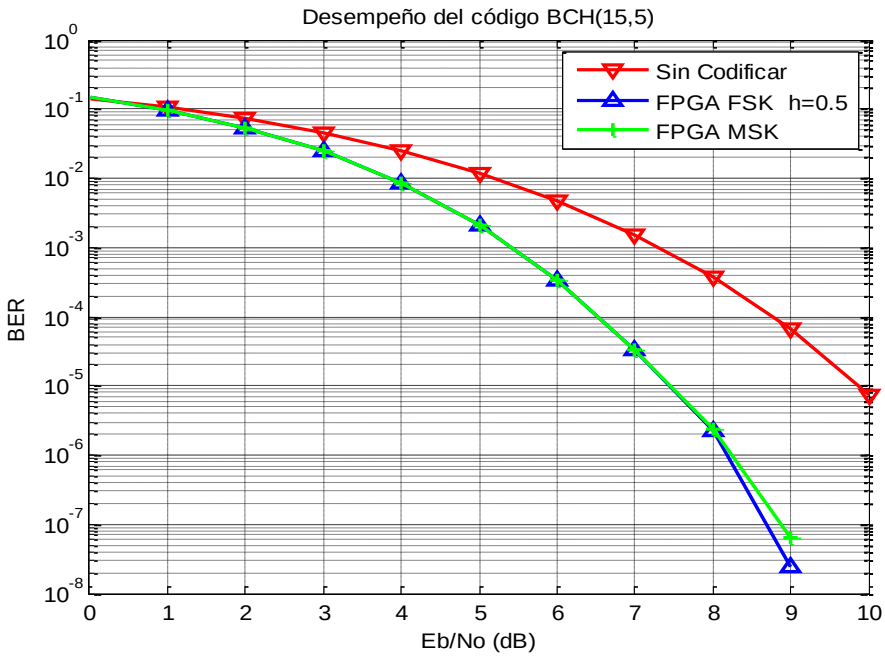


Figura 3.22 Comparación de la curva de desempeño del código $BCH(15,5)$ con modulación MSK y FSK, $h = 0.5$, Simulink, System Generator y FPGA.

3.2.5. Escenario 5

La figura 3.23 muestra las curvas de desempeño obtenidas para el sistema de comunicación banda base con codificación $BCH(31,26)$,$BCH(31,21)$ y $BCH(31,16)$ implementada con el esquema de modulación MSK en el FPGA para diferentes capacidades correctoras (t). Se puede apreciar que t influye directamente en su desempeño, debido a que el sistema puede corregir mayor número de errores.

Sin embargo, si se hace referencia al parámetro de eficiencia $\eta = k/n$, se puede inferir que el código $BCH(31,26)$ es más eficiente $(\eta = 0.838)$ que el código $BCH(31,16)$ $(\eta = 0.516)$, debido a que dentro de su palabra código contiene una palabra dato de mayor longitud $(26 > 16)$ y por lo tanto se tendrá una mayor tasa de información de la fuente.

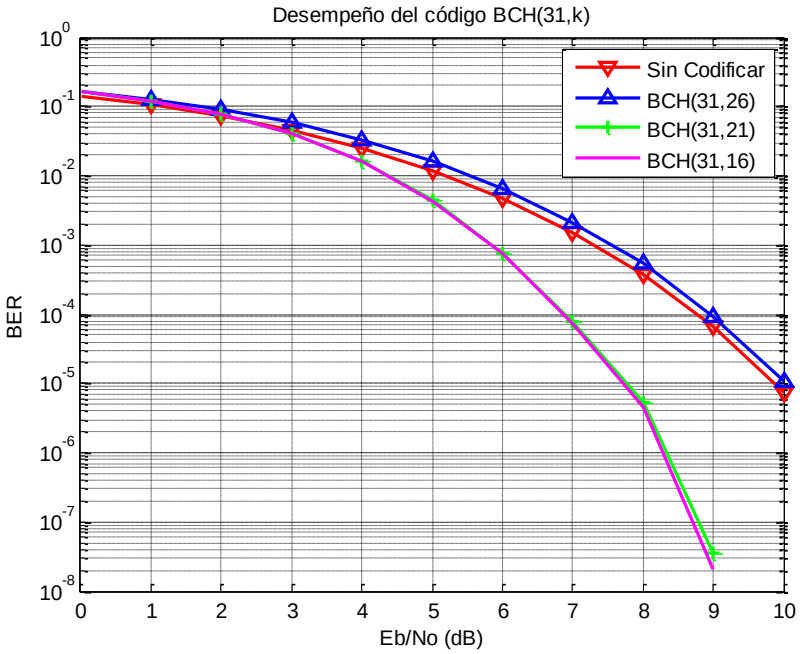


Figura 3.23 Comparación de la curva de desempeño del código $BCH(31,26)$, $BCH(31,21)$ y $BCH(31,16)$ con modulación MSK.

Por último es importante mencionar que el tamaño de la información (k) también interviene en el desempeño del sistema; entre menor sea la longitud del dato hay una menor incidencia en la decodificación de tipo errónea[18] y fallida[19]. Por ejemplo el código $BCH(31,16)$ presenta un mejor desempeño que el código $BCH(31,26)$; debido a que tiene 65536 (2^{16}) palabras mientras que el otro contiene 67108864 (2^{26}).

[18] Decodificación de tipo errónea: Corresponden a las palabras erróneamente decodificadas.
[19] Decodificación de tipo fallidas: Corresponden a palabras indecodificables.

En la sección D.2 del apéndice D, se encuentran las tablas con los promedios obtenidos de 20 repeticiones simuladas e implementadas de Simulink, System Generator y FPGA.

3.3. COMPARACIÓN DE RECURSOS HARDWARE

Las figuras 3.24, 3.25 y 3.26 presentan una comparación porcentual de los recursos utilizados para el sistema de comunicación implementado en el FPGA por los códigos $\mathrm{BCH}(7,4)$, $\mathrm{BCH}(15,k)$ y $\mathrm{BCH}(31,k)$ con los esquemas de modulación FSK con $h = 0.5$ y MSK. Para ello, se utilizan las siguientes convenciones:

1. Number of Slice Flip Flops
2. Number of 4 input LUTs
3. Number of occupied Slices
4. Number of Slices containing only related logic
5. Total Number of 4 input LUTs
6. Number of bonded IOBs
7. Number of BUFGMUX's
8. Number of MULT18X18SIOs
9. Number of RAMB 168WEs

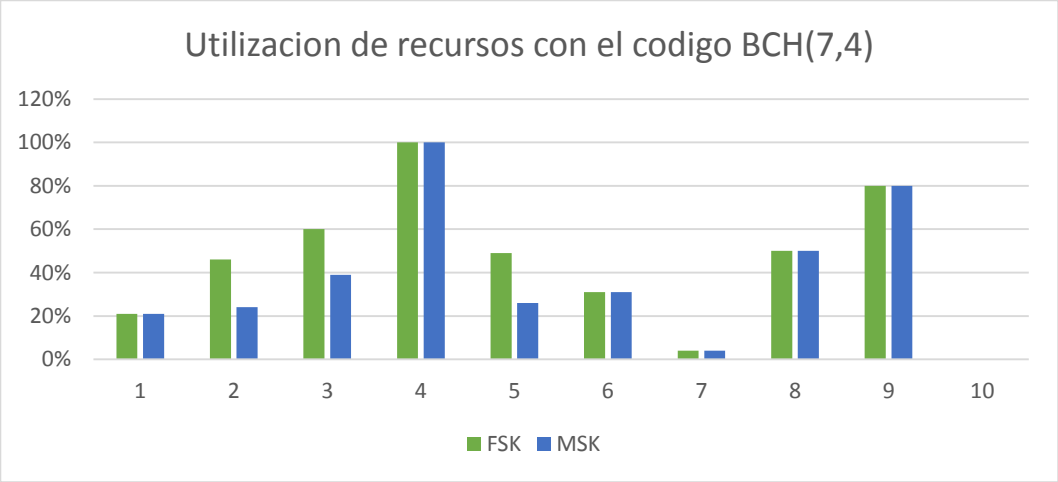


Figura 3.24 Comparación de los recursos utilizados del FPGA por el código $BCH(7,4)$ con modulación FSK y MSK

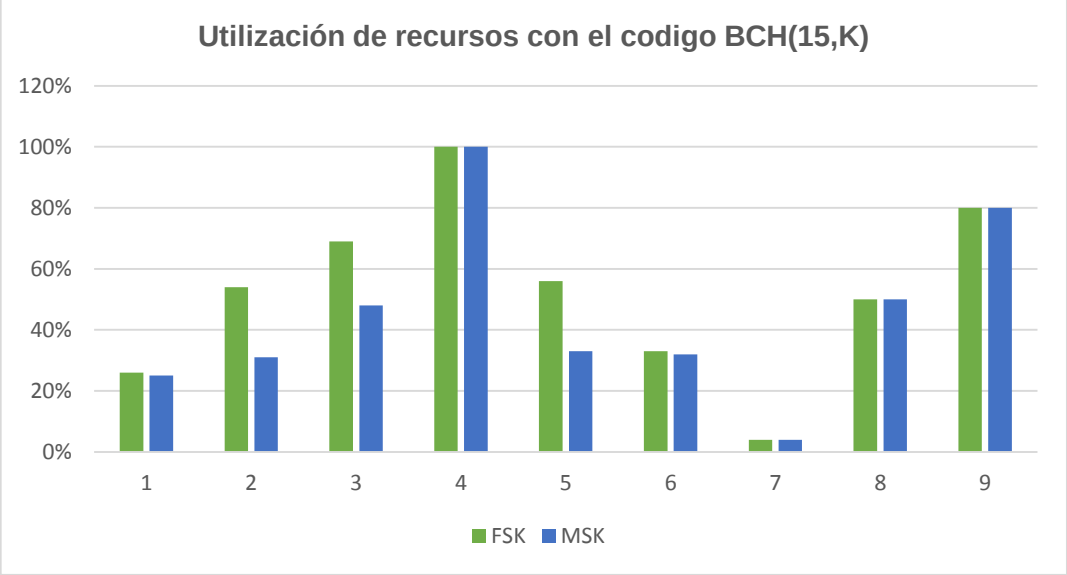


Figura 3. 25 Comparación de los recursos utilizados del FPGA por el código $BCH(7,4)$ con modulación FSK y MSK

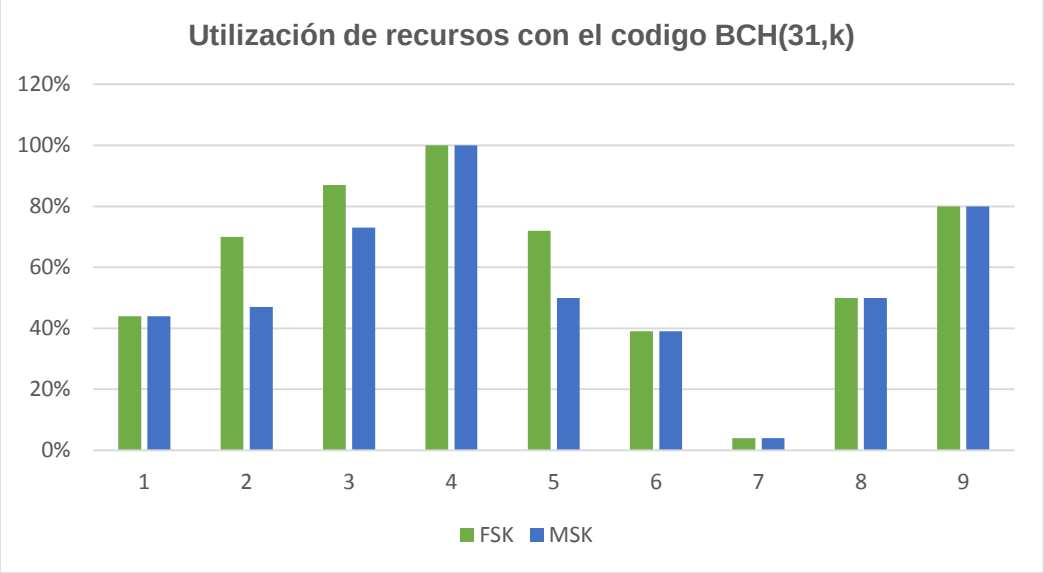


Figura 3.26 Comparación de los recursos utilizados del FPGA por el código $BCH(7,4)$ con modulación FSK y MSK

De lo anterior, se puede observar que la codificación $BCH(31,k)$ con FSK, $h = 0.5$ utiliza más recursos del FPGA debido a su longitud de código, al hacer uso de más Slice y LUT's en la etapa de decodificación; además se tiene en cuenta que este esquema de modulación utilizó más recursos en la fase 1, por lo cual al usar un código

$\mathrm{BCH}(31,26)$, $\mathrm{BCH}(31,21)$ y $\mathrm{BCH}(31,16)$ con FSK el manejo de recursos es mayor. En el apéndice E, se hace una descripción más detallada de los recursos utilizados.

3.3.1. Tiempo de uso de recursos por bit.

En las tablas 3.2, 3.3 y 3.4, se observa el reporte de tiempos de uso (*Timing Constraint*) arrojados por la herramienta Project Navigator de ISE; el dato más relevante es la restricción de tiempo calculado para la señal de reloj que controla el sistema de comunicación.

En la tabla 3.2 está consignada las restricciones de reloj para los códigos $BCH(15,k)$.

Tabla 3.2 Restricciones de Reloj para la codificación $BCH(15,k)$.

Esquema de modulación	Restricción del periodo de reloj (ns)		
	$(15,11)$	$(15,7)$	$(15,5)$
FSK, $h=0.5$	16,602	18,995	19,435
FSK, $h=0.25$	17,165	19,015	19,772
MSK	17,080	18,362	19,313

Tabla 3.3 Restricciones de Reloj para la codificación $BCH(31,26)$, $BCH(31,21)$ y $BCH(31,16)$

Esquema de modulación	Restricción del periodo de reloj (ns)		
	$(31,26)$	$(31,21)$	$(31,16)$
FSK, $h=0.5$	19,839	19,913	19,898
FSK, $h=0.25$	19,924	19,706	19,940
MSK	19,710	19,886	19,898

Tabla 3.4 Restricciones promedio de Reloj para los códigos $BCH(7,4)$, $BCH(15,k)$, $BCH(31,26)$, $BCH(31,21)$ y $BCH(31,16)$.

Esquema de modulación	Restricción del periodo de reloj (ns)		
	$(7,4)$	$(15,k)$	$(31,k)$
FSK, $h=0.5$	16,884	18,344	19,883
FSK, $h=0.25$	15,332	18,650	19,856
MSK	15,775	18,252	19,831

Se evidencia que los códigos $BCH(7,4)$, $BCH(31,26)$ y $BCH(31,k)$ utilizan aproximadamente el mismo tiempo de procesamiento requerido por el FPGA; sin embargo para códigos de longitud mínima existe una reducción aproximada del 3%, esta diferencia se relaciona con los recursos ocupados por el FPGA.

CAPÍTULO 4

CONCLUSIONES, RECOMENDACIONES Y TRABAJOS FUTUROS

4.1. CONCLUSIONES

Conforme a lo desarrollado y analizado en el presente trabajo de grado, se puede concluir lo siguiente:

- Las herramientas hardware y software utilizadas para la implementación de la codificación BCH binaria fueron adecuadas en cuanto a capacidad y velocidad de procesamiento; los recursos que ofrece la tarjeta Spartan 3A de Xilinx son suficientes para soportar los algoritmos de codificación y decodificación.

- A nivel de implementación, el proceso de decodificación es el que más recursos de la tarjeta Spartan 3A utiliza, debido a que los algoritmos ejecutados para hallar el síndrome, el polinomio localizador de error y el patrón de error son más complejos que los de la codificación.

- Los recursos utilizados por el FPGA son proporcionales a la longitud del código BCH binario y al esquema de modulación, de esta forma el código $\mathrm{BCH}(31{,}26)$ con modulación FSK ocupa más recursos que un $\mathrm{BCH}(7{,}4)$ con modulación MSK, dado que para el primer código se requiere un mayor número de procesos de decodificación y demodulación; así, para la implementación de un código de mayor longitud que el $\mathrm{BCH}(31{,}26)$ se requiere el uso de otro FPGA.

- El desempeño de los sistemas de comunicación banda base, implementados con codificación BCH binaria $\mathrm{BCH}(15{,}7)$, $\mathrm{BCH}(15{,}5)$, $\mathrm{BCH}(31{,}21)$ y $\mathrm{BCH}(31{,}16)$ con modulación FSK, $h = 0.5$ y $h = 0.25$, presenta en promedio una ganancia de codificación de $2{,}0725\ dB$ para una BER de 10^{-4} frente al sistema sin codificar.

- El desempeño de un sistema de comunicación banda base con codificación BCH binaria y modulación FSK, depende directamente de la capacidad correctora t de los códigos implementados. El mejor desempeño se obtuvo cuando $t = 3$, es decir con los códigos $\mathrm{BCH}(15,5)$ y $\mathrm{BCH}(31,16)$.

- El mejor desempeño para los códigos BCH binarios con $t = 3$ implementados en el FPGA se obtuvo para la modulación FSK con índice de modulación $h = 0.5$ que coincide con el desempeño de MSK.

4.2. RECOMENDACIONES

- En el momento de configurar la licencia para utilizar System Generator es importante generarla con la versión más actual del software, para que pueda crear el archivo .xise o .bit.

- Para un computador con sistema operativo de 64 bits es necesario realizar una configuración extra para que la herramienta iMPACT reconozca la tarjeta FPGA conectada al computador, lo que se recomienda para llevar a cabo este proceso es: copiar el archivo libPortabilityNOSH.dll alojado en la carpeta de Xilinx en la dirección “directorio_raiz_xilinx\version\ISE_DS\ISE \lib\nt64” por el archivo libPortability.dll ubicado en “directorio_raiz_xilinx \version\ISE_DS\ISE\lib\nt64”.

- En el momento de compilar los archivos en Simulink se debe repetir este proceso dos veces para que el software reconozca las variables configuradas.

- Cuando se desee trabajar con la aplicación Serial Communication, es importante primero que el computador reconozca la FPGA antes de conectarla al puerto serial.

4.3. TRABAJOS FUTUROS

- Analizar el desempeño de códigos BCH binarios con mayor longitud y capacidad correctora sobre otro tipo de hardware reconfigurable.

- Analizar el desempeño de la codificación BCH binaria con otros esquemas de modulación.

- Integrar la codificación BCH binaria con modulación FSK y MSK con un codificador de fuente.

BIBLIOGRAFÍA

[1] Genera Tecnologías “Ingeniería FPGA: Aplicaciones del FPGA”. [En linea]. Disponible: http://www.generatecnologias.es/aplicaciones_fpga.html

[2] M. E. Marin, J. K. Daza, "Análisis del desempeño de un sistema de comunicaciones con modulación MSK/FSK basado en hardware reconfigurable",Tesis de Pregrado, Dept. Telecomunicaciones, Universidad del Cauca, Popayán, Colombia, 2014.

[3] S. Lin y D. J. Costello, *Error Control Coding Fundamental and Applications*, F.F. Kuo,Ed. Pearson Prentice-Hall: New Jersey, 1983.

[4] A. Bateman, "Codificación de Canal", in *Comunicaciones digitales para el mundo real*, C.P. Civit, Ed. Marcombo: España,1999, pp. 173-178.

[5] W. Bolaños, D. Díaz, “Análisis del Desempeño de la Codificación Reed-Muller en un Canal con Ruido Blanco Aditivo Gaussiano”, Tesis de Pregrado, Dept. Telecomunicaciones, Universidad del Cauca, Popayán, Colombia, 2015.

[6] J. M. Ramírez, “Capítulo 5. Codificación de Control de Errores”, Material Académico. Departamento de Telecomunicaciones, Universidad del Cauca, Popayán, Colombia 2013.

[7] I.F. Blake, et al. "Introduction to Finite Fields and Bases", in *Applications of Finite Fields*, A.J Menezes, Ed. Springer:Canada,1993, pp. 1-15.

[8] L. W. Couch, "Introduction", in *Sistemas de Comunicaciones Digitales y Analógicos*, J.L. Cuevas, Ed. Pearson Prentice-Hall:México, 2008,pp. 19-29.

[9] R. Alvarado, "Códigos para detección y Corrección de errores en comunicaciones digitales", proc. *Revista de Ingenierías,* vol. 25, pp. 51-60, 2004.

[10] R. Acosta, "Estudio teórico práctico de los códigos no binarios de reeds-solomon para la detección y correción multiple de errores utilizando el método matricial", Tesis de especialización, Dep. Electrónica, Escuela Politecnica Nacional, Quito, Ecuador, 1994.

[11] L. Massey, "Shift-register synthesis and bch decoding", proc. *IEEE,* vol. 15, pp. 122-127, 1969.

[12] F. I. Peralta, et al. "Linear Complexy and the Berlekamp-Massey Algorithm in the Construction of Pseudorandom Sequence Generators" proc. *Información tecnológica,* vol. 17, pp. 167-178, 2006.

[13] B. Zeidman, "The Universal Design Methodology-taking hardware from conception" proc. *Edn Network,*pp. 53-55, 2006.

[14] National Instrument, "Introducción a la Tecnología FPGA: Los Cinco Beneficios Principales", [En línea]. Colombia, 2011. Disponible: http://www.ni.com/white-paper/6984/es/.

[15] I.Dominguez. J. J. Murillo, “Laboratorio de Comunicaciones Digitales Radio Definida por Software” Dep. Teoría de la Señal y Comunicaciones, Universidad de Sevilla, España, 2011.

[16] Prisma Software Gestión, “el blog del software de gestión en la pyme”, [En línea]. España, 2012. Disponible: http://www.prismasoftwaregestion.com/blog/2012-el-ano-de-los-miniordenadores-arm-de-bajo-coste/

[17] MathWorks, "Raspberry Pi Support from MATLAB", [En línea]. Estados Unidos, 2014. Disponible: http://www.mathworks.com/hardware-support/raspberry-pi-matlab.html.

[18] Arduino, “¿Que es Arduino?”, [En línea]. Santiago de Chile. Disponible: http://arduino.cl/que-es-arduino/.

[19] Mathworks, "Arduino Support from MATLAB", [En línea]. Estados Unidos, 2014. Disponible: http://www.mathworks.com/hardware-support/arduino-matlab.html.

[20] MathWorks, "Matlab, The language of Technical Computing", [En línea]. Estados Unidos, 2014. Disponible: http://www.mathworks.com/products/matlab/.

[21] D. A. H. Aponte, "Introducción a GNU O a GNU Octave", [En línea]. España, 2007. Disponible: http://softlibre.unizar.es/manuales/aplicaciones/octave/octave.pdf.

[22] "Open source software for numerical computation", [En línea]. Estados Unidos, 2015. Disponible: http://www.scilab.org/scilab/about.

[23] J. A. Muñoz, J. C. Zemanate, “Análisis del desempeño de un sistema de comunicaciones con modulación 16/64 QAM basado en hardware reconfigurable” Tesis de Pregrado, Dept. Telecomunicaciones, Universidad del Cauca, Popayán, Colombia, 2014.

[24] Xilinx All Programable “Software Development”, [En línea]. Estados Unidos, 2015. Disponible: http://www.xilinx.com/products/design-tools/vivado.html#documentation.

[25] V. Quintero y P. E. Jojoa, "Diseño e implementación de un prototipo decomunicación de datos basado en hardware reconfigurable Fase 1", Proyecto de Investigación, Dept. Telecomunicaciones, Universidad del Cauca, Popayan, Colombia, 2013.

[26] K.Saluja, "Linear Feedback Shift Registers Theory and Applications", Department of Electrical and Computer Engineering, University of Wisconsin-Madison, October 1991.

[27] O. M. Ulgen, J. J. Black, B. Johnsonbaugh y R. Klungle, “Simulation Methodology -A Practitioners Perspective,” Michigan Simulation User Group, Technical Committee on Simulation Methodology, 2000.

[28] C.Desset, et al. "Computing the Word, Symbol, and Bit-Error Rates for Block Error-Correcting Codes" proc. *IEEE,* vol. 52, pp. 910-921, 2004.

ANEXOS

Anexo A. Polinomio irreducible $\boldsymbol{f(x)}$ y representaciones polinomiales en el campo de Galois.

En la tabla A.1 se muestra un ejemplo del polinomio irreducible $f(x) = x^4 + x + 1$**.**

Tabla A.1 Polinomio irreducible en $GF(2)$.

$\boldsymbol{f(x)}$	$\boldsymbol{g(x)}$	$\boldsymbol{\frac{f(x)}{g(x)}}$	**Residuo**
$\boldsymbol{x^4 + x + 1}$	x	$x^3 + 1$	1
$\boldsymbol{x^4 + x + 1}$	$x + 1$	$x^3 + x^2 + 1$	1
$\boldsymbol{x^4 + x + 1}$	x^2	x^2	$x + 1$
$\boldsymbol{x^4 + x + 1}$	$x^2 + 1$	$x^2 + 1$	x
$\boldsymbol{x^4 + x + 1}$	$x^2 + x$	$x^2 + x + 1$	1
$\boldsymbol{x^4 + x + 1}$	$x^2 + x + 1$	$x^2 + x$	1
$\boldsymbol{x^4 + x + 1}$	x^3	x	$x + 1$
$\boldsymbol{x^4 + x + 1}$	$x^3 + 1$	x	1
$\boldsymbol{x^4 + x + 1}$	$x^3 + x$	x	$x^2 + x + 1$
$\boldsymbol{x^4 + x + 1}$	$x^3 + x^2$	$x + 1$	$x^2 + x + 1$
$\boldsymbol{x^4 + x + 1}$	$x^3 + x + 1$	x	$x^2 + 1$
$\boldsymbol{x^4 + x + 1}$	$x^3 + x^2 + 1$	$x + 1$	x^2
$\boldsymbol{x^4 + x + 1}$	$x^3 + x^2 + x$	$x + 1$	1
$\boldsymbol{x^4 + x + 1}$	$x^3 + x^2 + x + 1$	$x + 1$	x

Las representación polinomial de las raíces del campo $GF(2^3)$ y $GF(2^5)$ se pueden observar en las tablas A.2 y A.3.

Tabla A.2 Representación polinomial de las raíces del campo $GF(2^3)$ con polinomio primitivo $p(x) = x^3 + x + 1$.

Potencia	**Polinomio**	**Binaria**
0	0	000
1	1	001
$\boldsymbol{\alpha^1}$	α^1	010
$\boldsymbol{\alpha^2}$	α^2	100
$\boldsymbol{\alpha^3}$	$\alpha + 1$	011

$\boldsymbol{\alpha^4}$	$\alpha^2 + \alpha$	110
$\boldsymbol{\alpha^5}$	$\alpha^2 + \alpha + 1$	111
$\boldsymbol{\alpha^6}$	$\alpha^2 + 1$	101

Tabla A.3 Representación polinomial de las raíces del campo $GF(2^5)$ con polinomio primitivo $p(x) = x^5 + x^2 + 1$.

Potencia	Polinomio	Binaria
0	0	00000
1	1	00001
$\boldsymbol{\alpha^1}$	α	00010
$\boldsymbol{\alpha^2}$	α^2	00100
$\boldsymbol{\alpha^3}$	α^3	01000
$\boldsymbol{\alpha^4}$	α^4	10000
$\boldsymbol{\alpha^5}$	$\alpha^2 + 1$	00101
$\boldsymbol{\alpha^6}$	$\alpha^3 + \alpha$	01010
$\boldsymbol{\alpha^7}$	$\alpha^4 + \alpha^2$	10100
$\boldsymbol{\alpha^8}$	$\alpha^3 + \alpha^2 + 1$	01101
$\boldsymbol{\alpha^9}$	$\alpha^4 + \alpha^3 + \alpha$	11010
$\boldsymbol{\alpha^{10}}$	$\alpha^4 + 1$	10001
$\boldsymbol{\alpha^{11}}$	$\alpha^2 + \alpha + 1$	00111
$\boldsymbol{\alpha^{12}}$	$\alpha^3 + \alpha^2 + \alpha$	01110
$\boldsymbol{\alpha^{13}}$	$\alpha^4 + \alpha^3 + \alpha^2$	11100
$\boldsymbol{\alpha^{14}}$	$\alpha^4 + \alpha^3 + \alpha^2 + 1$	11101
$\boldsymbol{\alpha^{15}}$	$\alpha^4 + \alpha^3 + \alpha^2 + \alpha + 1$	11111
$\boldsymbol{\alpha^{16}}$	$\alpha^4 + \alpha^3 + \alpha + 1$	11011
$\boldsymbol{\alpha^{17}}$	$\alpha^4 + \alpha + 1$	10011
$\boldsymbol{\alpha^{18}}$	$\alpha + 1$	00011
$\boldsymbol{\alpha^{19}}$	$\alpha^2 + \alpha$	00110
$\boldsymbol{\alpha^{20}}$	$\alpha^3 + \alpha^2$	01100
$\boldsymbol{\alpha^{21}}$	$\alpha^4 + \alpha^3$	11000
$\boldsymbol{\alpha^{22}}$	$\alpha^4 + \alpha^2 + 1$	10101
$\boldsymbol{\alpha^{23}}$	$\alpha^3 + \alpha^2 + \alpha + 1$	01111
$\boldsymbol{\alpha^{24}}$	$\alpha^4 + \alpha^3 + \alpha^2 + \alpha$	11110
$\boldsymbol{\alpha^{25}}$	$\alpha^4 + \alpha^3 + 1$	11001
$\boldsymbol{\alpha^{26}}$	$\alpha^4 + \alpha^2 + \alpha + 1$	10111
$\boldsymbol{\alpha^{27}}$	$\alpha^3 + \alpha + 1$	01011
$\boldsymbol{\alpha^{28}}$	$\alpha^4 + \alpha^2 + \alpha$	10110
$\boldsymbol{\alpha^{29}}$	$\alpha^3 + 1$	01001
$\boldsymbol{\alpha^{30}}$	$\alpha^4 + \alpha$	10010

Anexo B. Polinomios mínimos de los elementos del campo de Galois.

En las tablas B.1, B.2 y B.3 se observan los polinomios mínimos de los elementos correspondientes a $GF(2^3)$, $GF(2^4)$ y $GF(2^5)$ respectivamente.

Tabla B.1 Polinomios mínimos del campo $GF(2^3)$.

Raíces conjugadas	Polinomio mínimo
$\mathbf{0}$	x
$\mathbf{1}$	$x+1$
$\boldsymbol{\alpha}$, $\boldsymbol{\alpha^2}$, $\boldsymbol{\alpha^4}$	x^3+x+1
$\boldsymbol{\alpha^3}$, $\boldsymbol{\alpha^5}$, $\boldsymbol{\alpha^6}$	x^3+x^2+1

Tabla B.2 Polinomios mínimos del campo $GF(2^4)$.

Raíces conjugadas	Polinomios mínimos
$\mathbf{0}$	x
$\mathbf{1}$	$1+x$
$\boldsymbol{\alpha}$, $\boldsymbol{\alpha^2}$, $\boldsymbol{\alpha^4}$, $\boldsymbol{\alpha^8}$,	x^4+x+1
$\boldsymbol{\alpha^3}$, $\boldsymbol{\alpha^6}$, $\boldsymbol{\alpha^9}$, $\boldsymbol{\alpha^{12}}$,	$x^4+x^3+x^2+x+1$
$\boldsymbol{\alpha^5}$, $\boldsymbol{\alpha^{10}}$	x^2+x+1
$\boldsymbol{\alpha^7}$, $\boldsymbol{\alpha^{11}}$, $\boldsymbol{\alpha^{13}}$, $\boldsymbol{\alpha^{14}}$	x^4+x^3+1

Tabla B.3 Polinomios mínimos del campo $GF(2^5)$.

Raíces conjugadas	Polinomio mínimo
$\mathbf{0}$	x
$\mathbf{1}$	$x+1$
$\boldsymbol{\alpha}$, $\boldsymbol{\alpha^2}$, $\boldsymbol{\alpha^4}$, $\boldsymbol{\alpha^8}$, $\boldsymbol{\alpha^{16}}$	x^5+x^2+1
$\boldsymbol{\alpha^3}$, $\boldsymbol{\alpha^6}$, $\boldsymbol{\alpha^{12}}$, $\boldsymbol{\alpha^{17}}$, $\boldsymbol{\alpha^{24}}$	$x^5+x^4+x^3+x^2+1$
$\boldsymbol{\alpha^5}$, $\boldsymbol{\alpha^9}$, $\boldsymbol{\alpha^{10}}$, $\boldsymbol{\alpha^{18}}$, $\boldsymbol{\alpha^{20}}$	$x^5+x^4+x^2+x+1$
$\boldsymbol{\alpha^7}$, $\boldsymbol{\alpha^{14}}$, $\boldsymbol{\alpha^{19}}$, $\boldsymbol{\alpha^{25}}$, $\boldsymbol{\alpha^{28}}$	$x^5+x^3+x^2+x+1$
$\boldsymbol{\alpha^{11}}$, $\boldsymbol{\alpha^{13}}$, $\boldsymbol{\alpha^{21}}$, $\boldsymbol{\alpha^{22}}$, $\boldsymbol{\alpha^{26}}$	$x^5+x^4+x^3+x+1$
$\boldsymbol{\alpha^{15}}$, $\boldsymbol{\alpha^{23}}$, $\boldsymbol{\alpha^{27}}$, $\boldsymbol{\alpha^{29}}$, $\boldsymbol{\alpha^{30}}$	x^5+x^3+1

Anexo C. Diseño para el proceso de la división polinomial.

Para llevar a cabo este proceso, se debe construir un circuito de LFSR's como se ve en la figura C.1, donde se dividirá una entrada polinomial $p(x)$ por $g(x)$.

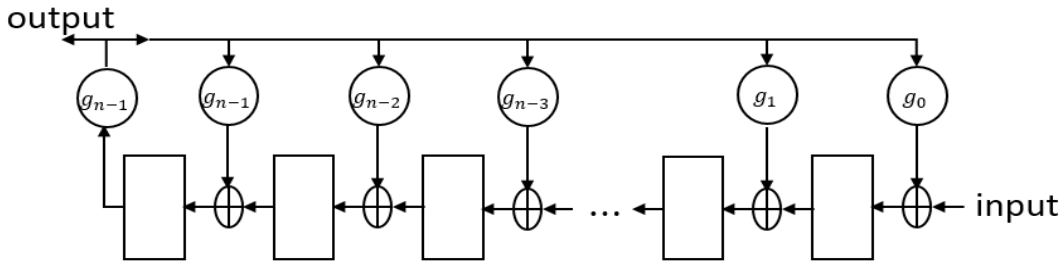


Figura C.1 LFSR para dividir un polinomio $g(x)$.

La figura C.2 muestra un circuito de LFSR que divide por $x^5 + x^4 + x^2 + 1$. Inicialmente los flip-flops se establecen en cero. La salida es cero para el primer cambio n hasta que el primer símbolo de entrada alcance el final del registro. La primera salida distinta a cero es $q_r g_n^{-1} = q_r$, cuando se está calculando $q(x)/g(x)$. Para cada coeficiente h_j, el polinomio $h_j g(x)$ debe ser restado del dividendo. Después de un total de r cambios todo el cociente se ha desplazado hacia afuera, y el resto se mantiene en el registro de desplazamiento.

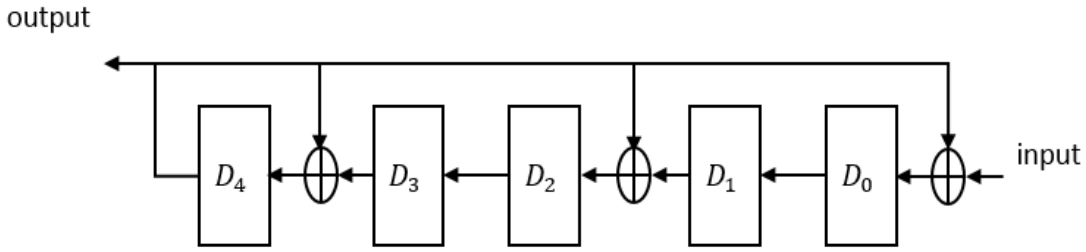


Figura C.2 LFSR que divide por $g(x) = x^5 + x^4 + x^2 + 1$.

La figura C.3 muestra un ejemplo de los cambios de los registros para $q(x)/g(x) = (x^7 + x^6 + x^5 + x^4 + x^2 + 1) \ /(x^5 + x^4 + x^2 + 1)$ y su equivalente expresión matemática en la figura C.4.

Salida de datos	D_4	D_3	D_2	D_1	D_0	Entrada de datos
	0	0	0	0	0	1 1 1 1 0 1 0
						1
1	1	1	1	1	0	1 0 A
1 0	0	1	0	0	0	0 1 B
1 0 1	1	0	0	0	0	1
1 0 1 1	1	0	1	0	0	C

Figura C.3 Cómputo de los LFSR de $q(x)/g(x)$.

$$
\begin{array}{l|l}
x^7+x^6+x^5+x^4+x^2+1 \quad \text{A} & x^5+x^4+x^2+1 \\
\cline{2-2}
x^7+x^6 \quad\quad +x^4+x^2 & x^2+1 \\
\cline{1-1}
x^5 \quad\quad\quad +1 \quad \text{B} & \\
x^5+x^4+x^2+1 & \\
\cline{1-1}
x^4+x^2 \quad\quad \text{C} &
\end{array}
$$

Figura C.4 Expresión matemática de la división $q(x)/g(x)$.