PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

Estudio de una arquitectura para corrección de errores mediante códigos Bose-Chaudhuri-Hocquenghem (BCH) para aplicaciones de nano-satélites

TRABAJO DE INVESTIGACIÓN PARA LA OBTENCIÓN DEL GRADO DE BACHILLER EN CIENCIAS CON MENCIÓN EN INGENIERÍA ELECTRÓNICA

Mayte Rociel Giraldo Solis

ASESOR: Mg. Ing. Mario Andrés Raffo Jara

Lima, 2020

Resumen

En la comunicación satelital se transmiten datos los cuales pueden verse afectados por diversos factores como la radiación. Por esta razón, el Comité Consultivo para Sistemas de Datos Espaciales (CCSDS por sus siglas en inglés) y la Cooperación Europea para Estandarización Espacial (ECSS por sus siglas en inglés) recomiendan la implementación de códigos que permitan corregir dichos errores. Además, especifican el uso del código Bose-Chaudhuri-Hoquenghem (BCH) frente a Reed-Solomon (RS) debido a su capacidad de corrección de multiples errores y que esto se realiza bit a bit, es decir, no importa la posición del error. Se recomienda el uso del código BCH (63,56), ya que permite corregir 1 bit errado y detectar 2, suficiente para ser implementado en un nanosatélite. Dicho código hace referencia a 56 bits para información o datos y 7 bits para el control de errores, con lo que se obtiene un total de 63 bits.

El decodificador BCH consta de 3 bloques: cálculo de síndromes, localización del error y búsqueda de Chien. El bloque de mayor relevancia es el cálculo de síndrome, debido a que este permite conocer si la palabra a decodificar contiene error, y de ser así si es posible realizar la corrección. Mientras que los otros bloques usan los síndromes hallados para encontrar la posición del error.

En este trabajo, se presenta un estudio del diseño de una arquitectura de un decodificador para correccion de errores mediante el código BCH (63,56), asi como las consideraciones para cada uno de los bloques obteniéndose el modelo de solución.

Índice General

Índice de Figuras

Índice de Tablas

Introducción

Las comunicaciones satelitales cada día son más necesarias en diversos ámbitos como la transmisión para televisión, el uso del GPS y el estudio de la luna, el sol, asteroides, entre otros por parte de agencias espaciales como la NASA y universidades como es el caso de la PUCP. Los datos transmitidos pueden verse afectados por diversos factores espaciales, por lo que es necesaria la implementación de una técnica que pueda detectar los errores introducidos. La técnica idónea para la comunicación satelital es la FEC, debido a su capacidad de corregir errores. En esta técnica se tienen diversos códigos como Reed-Solomon (RS) y Bose-Chaudhuri-Hoquenghem (BCH). Sin embargo, estos últimos son los recomendados por el Comité Consultivo para Sistemas de datos Espaciales (CCSDS por sus siglas en inglés) y la Cooperación Europea para la Estandarización Espacial (ECSS por sus siglas en inglés), los que son organizaciones encargadas de presentar protocolos en el diseño de sistemas de comunicación satelital. Con los códigos BCH se tiene la posibilidad de una corrección de errores múltiple y sin importar las posiciones de estos; por ello, serán estos los que se estudiarán.

En el presente trabajo se realizará un estudio de una arquitectura adecuada, en términos de eficiencia, para el diseño de un decodificador BCH. En el capítulo 1 se presenta el marco problemático, la justificación del trabajo de investigación y los objetivos del mismo. En el capítulo 2 se realiza el estudio del decodificador BCH y los fundamentos necesarios para su entendimiento, así como una revisión de las arquitecturas que podrían implementarlas como modelo de solución. Finalmente, se muestran las conclusiones, recomendaciones y trabajos futuros.

Capítulo 1

Marco problemático de errores en transmisiones de satélites

En la actualidad, la transferencia de datos está cumpliendo un papel importante, debido al incremento de la necesidad por transmitir digitalmente información (comunicación de datos digitales, comunicaciones móviles y satelitales).

En los 10 últimos años ha habido avances que indicarían el comienzo de una nueva era de exploración espacial [10], por lo que la comunicación satelital, se vuelve fundamental. De acuerdo con la NASA, los temas de investigación conciernen a la luna, al sol, a asteroides, entre otros, dentro de ellos se tienen diversos objetivos como caracterizar los efectos de la radiación sobre organismos vivos y apoyar la investigación del clima espacial. No obstante, a medida que pasan los años se busca llegar más lejos, explorar el espacio profundo, por ello, para el presente año se tiene el lanzamiento de la misión de exploración 1 (del inglés Exploration Mission 1, nemónico EM-1), el cual proporcionará una base para una posterior exploración humana del espacio profundo [10], [11].

Se debe destacar que, para el cumplimiento de los objetivos mencionados en el anterior párrafo, es sustancial la implementación de satélites que posibiliten la transferencia de información obtenida hacia la estación base. Es necesario indicar que debido a las dificultades de traslado de peso, la NASA, la industria y algunos centros educativos prefieren el uso de nanosatélites que, a diferencia de los satélites medianos (comúnmente implementados por grandes compañías por el elevado financiamiento) tienen un peso que se encuentra en el rango de 1kg a 10kg, y las medidas se reducen a 10cm por lado como se muestra en la figura 1.1.

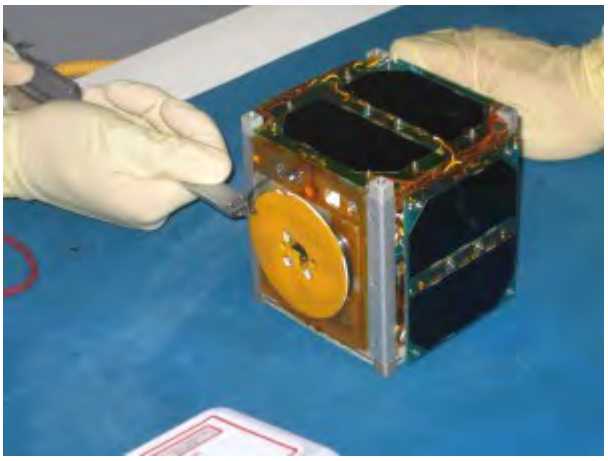

Figura 1.1: Cubo satélite PUCP-SAT-1 [1]

La implementación de los nanosatélites no solo abarca compañías, sino que su diseño actualmente es accesible a universidades. Por ejemplo, la Pontificia Universidad Católica del Perú (PUCP) en el año 2013 permitió ingresar al Perú en la era espacial [1], con el lanzamiento de PUCP-SAT-1, el primer satélite peruano. Para este proyecto fue necesaria la implementación de un software de control y seguimiento, además de la estación terrestre.

1.1. Técnicas de corrección de errores

En la comunicación satelital, se debe cumplir con obtener la información correcta en la estación base en Tierra. Sin embargo, debido a efectos ionosféricos o troposféricos, atenuación en la propagación de ondas de radio debido a la precipitación (lluvia, nieve, granizo, etc.), absorción gaseosa entre oxígeno y vapor de agua, entre otros, no es posible garantizar una transmisión confiable de información [12]. Por lo tanto, es necesaria la implementación de técnicas que permitan asegurar la precisión de datos en el receptor. Para cumplir con lo anterior, se debe implementar una de las 2 formas de detección de errores: consulta automática de repetición (del inglés Automatic Repeat Query, nemónico ARQ) y corrección de errores hacia adelante (del inglés Forward error correction, nemónico FEC).

Para una comunicación satelital usando el método ARQ, se deben colocar bits adicionales a los de datos que servirán para detectar errores. Entonces, cuando se envié información se tendrán 2 casos: paquete correcto y paquete incorrecto. Para el primer caso, el receptor enviará un ACK (del inglés acknowledgement, reconocimiento) para confirmar que los datos son correctos, mientras que para el segundo caso el receptor enviará NACK (del inglés negative acknowledgment, reconocimiento negativo), y pedirá la retransmisión del mensaje original hasta

que este sea el correcto y pueda enviar ACK. Por otro lado, para una comunicación satelital usando el método FEC, se usará redundancia de bits en el mensaje lo que permite no solo la detección de errores sin retransmisión, sino también la capacidad de corregir errores [13].

De lo anteriormente expuesto, se obtiene que para comunicaciones que implican grandes distancias no es conveniente el uso del primer método, ARQ, debido al tiempo que toma la retransmisión del paquete de datos, sino el segundo que permitiría la corrección de errores en la estación base.

1.2. Justificación del trabajo de investigación

Para el método FEC, se tiene diferentes códigos: Bose-Chaudhuri-Hocquenghem (BCH), Reed-Solomon (RS), Hamming, entre otros. No obstante, como se hace referencia en [14], los códigos más utilizados en el área de comunicación satelital son los códigos BCH y RSunoEn el capítulo 2 se presentarán los respectivos fundamentos teóricos., por lo que será necesario realizar una comparación entre estos para elegir el más adecuado para la aplicación.

1.2.1. Comparación entre códigos BCH y RS

En principio, los códigos BCH son códigos binarios, es decir, la detección y corrección de errores se realiza en información binaria o bits, mientras que los códigos RS (caso particular de los BCH) son no binarios, ya que se trabaja sobre símbolos, los cuales pueden contener más de 1 bit. De esto se obtiene que usando códigos BCH, una vez localizado el error lo único es cambiar el valor del bit; sin embargo, en el caso de los códigos RS no solo se debe obtener la ubicación sino también la magnitud del error para ser corregido. Adicionalmente, el BCH tiene la capacidad de corregir errores aleatorios, mientras que el RS es capaz de corregir ráfagas, con lo que se tiene una desventaja en caso se extienda sobre varios símbolos porque no podrá ser corregido, mientras que BCH sí podría hacerlo porque la posición de la ráfaga no depende del símbolo [15], [16]. La complejidad de los códigos BCH se concentra en su decodificación debido al uso de campos finitos o campos de Galois (los códigos RS también los usan), no obstante, mediante el uso de síndromesuno en códigos BCH se tiene mayor facilidad respecto a la decodificación en RS [13].

Cómite consultivo para sistemas de datos

El cómite consultivo para sistemas de datos (del inglés Consultative Committee for Space Data Systems, nemónico CCSDS) fue fundado en 1982 por las 11 principales agencias espaciales

como la NASA y UK Space Agency. Este es un comité encargado de presentar protocolos para el diseño de sistemas de comunicación satelital [17]. Para la técnica de control de errores, el comité recomienda la implementación de códigos BCH, debido a su corrección múltiple de errores sin importar la posición de ellos, además de su algoritmo de decodificación simple [2].

La figura 1.2 muestra el diagrama de bloques del envío de datos desde el satélite. La trama a enviarse debe ser codificada, en este caso se realiza mediante códigos BCH. El bloque opcional CLTU (del inglés Communication Link Transfer Unit, Unidad de transferencia de enlace de comunicación) es usado para sincronizar la trama, es decir, la secuencia de inicio con la de espera. Finalmente, el bloque PLOP (del inglés Physical Layer Operation Procedure, Procedimiento de operación de capa física) modula el código y envía las ondas de radio [2].

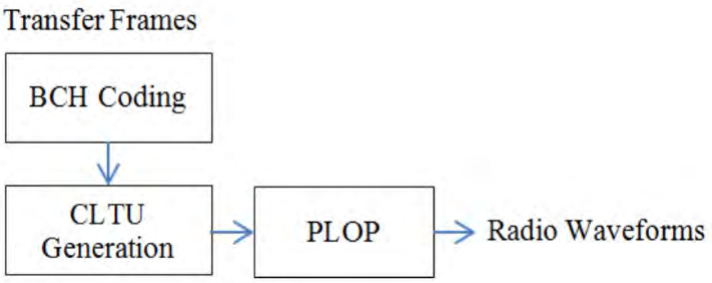

Figura 1.2: Diagrama de bloques del envío de datos [2]

La figura 1.3 representa la recepción de las ondas de radio de figura 1. El bloque Start Sequence Search (Iniciar búsqueda de secuencia) busca la secuencia de inicio generada anteriormente por el bloque CLTU. Esta secuencia ingresa a un decodificador BCH. El comando obtenido a la salida de este bloque ingresará a otro de decisión para ser aceptado o rechazado. En caso sea aceptado se envía la secuencia decodificada a la salida [2].

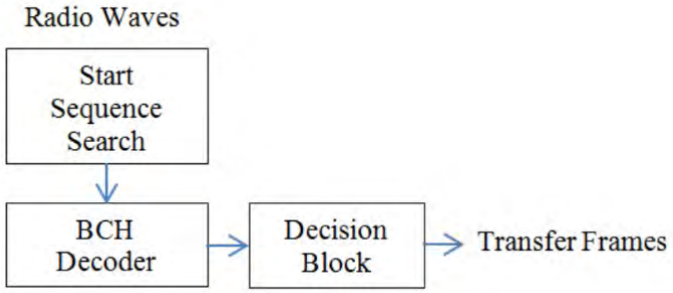

Figura 1.3: Diagrama de bloques de la recepción de datos en la estación base [2]

1.3. Objetivos del trabajo de investigación

1.3.1. Objetivo general

El objetivo general del trabajo de investigación es el estudio de una arquitectura de corrección de errores para un decodificador BCH, como modelo de solución.

1.3.2. Objetivos específicos

- Entender el funcionamiento interno del algoritmo de corrección de errores BCH.
- Estudiar los modelos matemáticos de Campos de Galois para el funcionamiento de los bloques del decodificador BCH.
- Estudiar el diseño de cada uno de los bloques del decodificador y la relación entre ellos para la corrección del error mediante BCH (63,56)

Capítulo 2

Fundamentos teóricos de corrección de errores

En el presente capítulo se presentarán los fundamentos teóricos de la elección de los códigos BCH, según lo mencionado en la sección 1.2.1 del capítulo 1. Además, se realizará una revisión de campos de Galois necesaria para diseñar el codificador y decodificador BCH. Finalmente, se efectuará una revisión de arquitecturas implementadas en el decodificador con el objetivo de establecer un modelo de solución.

2.1. Comparación entre códigos BCH y RS

Los códigos BCH son los más eficientes y potentes para la corrección de errores aleatorios y potentes, debido a su amplio rango de velocidad y capacidad de corrección de múltiples errores [18]. Estos son también llamados códigos binarios ya que su detección y corrección se realiza bit por bit. Por otro lado, códigos RS son denominados no binarios puesto que la detección y corrección se realiza por símbolos, los cuales pueden contener más de 1 bit.

En la decodificación, para localizar la ubicación del error, ambos códigos son similares por sus 3 bloques principales: syndrome computation (cálculo de síndromes), key equation solver (solucionador de ecuaciones clave) y Chien Search (búsqueda de Chien). Sin embargo, como se ilustra en la figura 2.1, los códigos RS necesitan un bloque adicional, Forney's algorithm, para determinar la magnitud del error, debido a que no necesariamente es un bit [3]. Este bloque adicional genera un incremento en el tiempo de cálculo para el hardware.

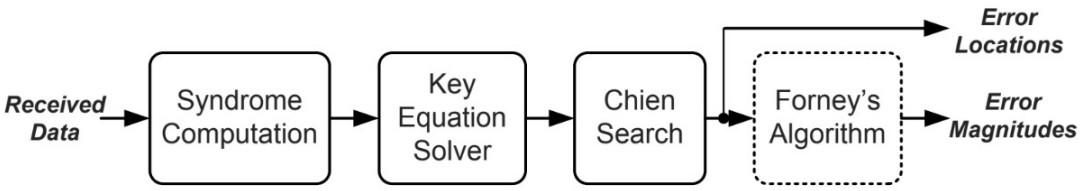

Figura 2.1: Localización de errores en decodificadores BCH/RS [3]

Las comunicaciones satelitales requieren un bloque de modulación con el que la información (datos) se transformen en ondas de radio y puedan ser enviadas. Esta modulación puede ser por medio de Quadrature Amplitude Modulation (Modulación de amplitud en cuadratura / QAM) o Phase Shift Keying (Modulación por desplazamiento de fase / PSK). Además, se debe considerar que en todo canal existe ruido [19].

En las figuras 2.2 y 2.3, se muestran la comparación de los códigos BCH y RS respecto a su rendimiento en un canal de desvanecimiento de Rayleigh para una modulación BPSK y QAM. Se puede observar que en ambos casos los códigos BCH son más eficientes que los RS por su bajo valor en BER (Bit error rate / Tasa de bit errados) [4].

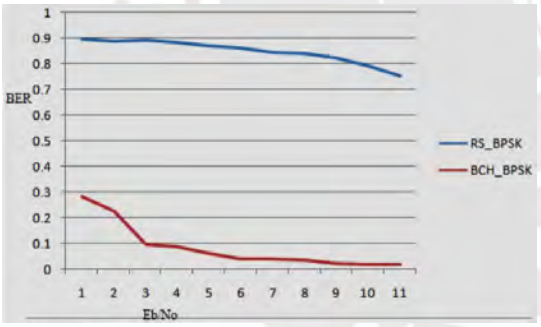

Figura 2.2: BCH y RS en modulación BPSK [4]

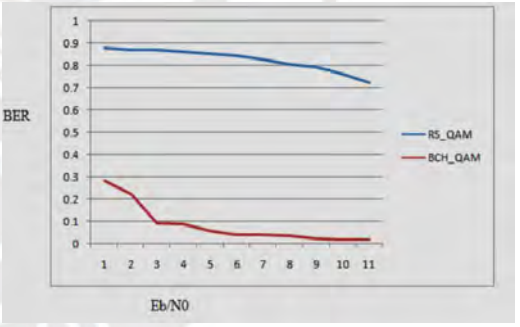

Figura 2.3: BCH y RS en modulación QAM [4]

En conclusión, debido a que las comunicaciones satelitales requieren minimizar el tiempo de cálculo y tener una eficiencia alta en cuanto a la tasa de bits errados, los códigos BCH son los adecuados y necesarios para la corrección de bits errados.

2.2. Campos de Galois, codificador y decodificador BCH

Los códigos BCH tienen como fundamento matemático a los campos finitos o campos de Galois. Para ello será necesario realizar una breve introducción.

2.2.1. Campos de Galois

Son representados como $GF(q)$, donde q es la potencia de un número primo, el número de elementos del campo y el orden del mismo. Se llama un campo de Galois binario cuando $q = 2$. $GF(2)$ son los campos más simples, aunque su importancia radica en que las comunicaciones digitales son binarias. Los elementos en el campo son $\{0, 1, 2, ..., q-1\}$. De manera general, los campos binarios son representados por $GF(2^m)$ [20].

Para efectuar las operaciones en campos de Galois, se debe considerar además la operación módulo de la siguiente manera. Se tomará el siguiente ejemplo:
- Sea el campo $GF(3)$

- Los elementos del campo son $\{0, 1, 2\}$.
- Se debe realizar las operaciones de suma y multiplicación entre sus elementos.
- Sumar o multiplicar los elementos de manera cotidiana.
- Dicho resultado parcial se debe dividir con el número de elementos del campo (3), de la cual se tendrá un residuo (operación módulo), este será el resultado final de la operación.

En la tabla 2.1, se muestran los resultados de la operación suma entre sus elementos, y se obtiene también que dicho resultado pertenece al campo. Esto es replicado en la tabla 2.2 para multiplicación.

Tabla 2.1: Operación suma

+	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

Tabla 2.2: Operación multiplicación

.	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

Los elementos pueden ser representados también de forma polinomial.

$$f(x) = f_{n-1}x^{n-2} + f_{n-2}x^{n-2} + ... + f_1x^1 + f_0 \tag{2.1}$$

Para el campo binario los valores de f_x solo pueden ser 0 o 1. Además, en este campo la suma coincide con la puerta lógica XOR bit a bit. Por ejemplo, para $1 + 1 = 2$, 2 mod 2 = 0, lo cual coincide con la puerta XOR $1 \oplus 1 = 0$, también par $1 + 0 = 1$, 1 mod 2 = 1, lo que es igual a $1 \oplus 0 = 1$ [21].

2.2.2. Codificador BCH (n,k)

En el capítulo 1, se hizo referencia al Comité consultivo para sistemas de datos espaciales (CCSDS por sus siglas en inglés), el cual tenía como objetivos brindar un estándar para el diseño de sistemas de comunicación satelital. En la técnica de control de errores no solo se recomienda el uso de códigos BCH, sino también emplear 56 bits para datos o información y 63 bits para su codificación [22] [2] [12] [5].

Para los códigos BCH se tienen los siguientes parámetros:

- k es el tamaño (número de bits) del mensaje que se desea enviar.
- n es el tamaño (número de bits) del mensaje codificado.
- t es la máxima cantidad de errores que se pueden corregir en el decodificador sin importar la posición de estos.
- d_{min} es la distancia de Hamming, la cual debe cumplir $d_{min} \geq 2t + 1$.

Los códigos BCH pueden ser implementados mediante una codificación sistemática o no sistemática, esto depende de cómo se decida codificar el mensaje.

Se debe transmitir un mensaje, el cual puede ser denotado en su forma polinomial $m(x) = m_0 + m_1x + m_2x^2 + +m_{k-1}x^{k-1}$, donde $m_0, m_1, m_2, \ldots, m_{k-1}$ representan los dígitos del mensaje. Además, el polinomio $g(x) = g_0 + g_1x + g_2x^2 + +g_{n-k-1}x^{n-k-1}x^{n-k}$, es denominado el polinomio generador, el cual se obtiene de escoger un polinomio primitivo de grado m y contruir $GF(2^m)$.

1. Codificación no sistemática: En este tipo de codificación, el mensaje $m(x)$ es un factor de la palabra codificada $c(x)$ en su forma más simple, mutiplicando al polinomio generador $g(x)$, como se muestra en la ecuación 2.2.

$$c(x) = m(x)g(x) \tag{2.2}$$

2. Codificación sistemática: En este tipo de codificación, la obtención del polinomio de la palabra código a enviarse $c(x)$ no se calcula directamente, sino mediante las operaciones de multiplicación y módulo, como se muestra en la ecuación 2.3

$$c(x) = p(x) + x^{n-k}m(x) \;\; donde \;\; p(x) = (x^{n-k}m(x))_{g(x)} \tag{2.3}$$

A pesar de la facilidad para hallar el polinomio de la palabra código $c(x)$ en la codificación no sistemática, se prefiere la implementación de codificación sistemática, debido a que el mensaje $m(x)$ es parte de $c(x)$ en sus últimos k bits [23].

En la figura 2.4, se muestra el diagrama de una codificación sistemática para el código propuesto por el Comité consultivo anteriormente mencionado. En este se puede observar que la información o mensaje es parte de la palabra o bloque de código [5].

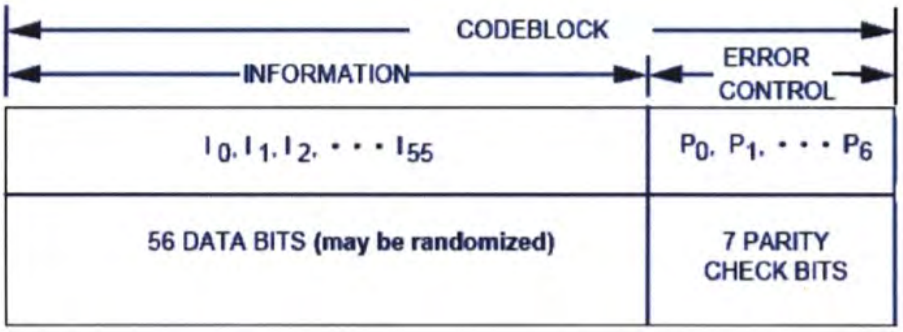

Figura 2.4: Diagrama de codificación sistemática para BCH (63,56) [5]

En la figura 2.5, se muestra el circuito de codifcación $BCH(n,k)$, donde $i(x)$ representa a $m(x), g_1, g_2, ..., g_{n-k-1}$ son los coeficientes de $g(x)$ y $b_0, b_1, ..., b_{n-k-1}$ representan registros de desplazamiento.

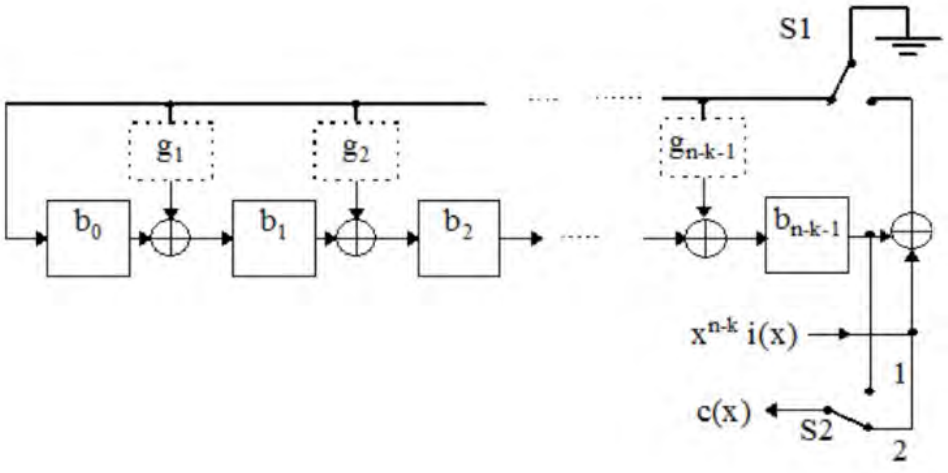

Figura 2.5: Circuito de codificación BCH (n,k) [6]

2.2.3. Decodificador BCH

Debido a los fenómenos mencionados en el capítulo 1, la palabra código transmitida por el codificador $c(x)$ no es la recibida en el decodificador, sino que a esta se le suma un error, el cual

puede ser expresado en su forma polinomial $e(x)$. La ecuación 2.4 representa el mensaje recibido a la entrada del decodificador [8].

$$r(x) = c(x) + e(x) \tag{2.4}$$

Se deben seguir los siguientes pasos para decodificar $r(x)$:

1. Almacenar en un registro el mensaje recibido $r(x)$
2. Calcular los síndromes
3. Determinar el polinomio localizador de errores
4. Hallar las raíces del polinomio localizador de errores

Estos pasos son mostrados en la figura 2.6, donde $i_{62}, i_{61}, \ldots, i_0$ representan los bits de $r(x), s_1, s_2, \ldots, s_{2t}$ son los síndromes calculados y $\Lambda_1, \Lambda_2, \ldots, \Lambda_v$ con $v \geq t$ son los coeficientes del polinomio localizador de errores.

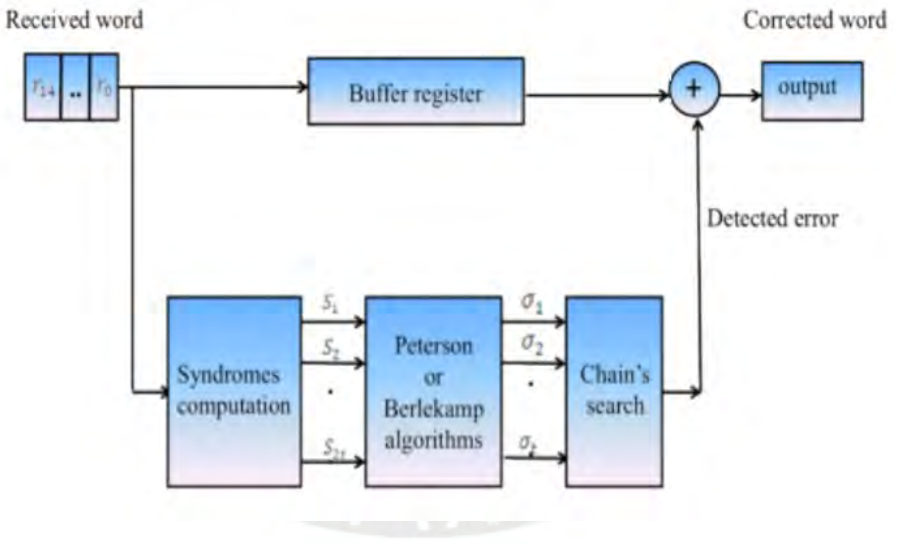

Figura 2.6: Diagrama de bloques del decodificador BCH [7]

- Se debe almacenar en un registro el mensaje recibido $r(x)$ como se muestra en la figura 2.6 con Buffer register, esto con el objetivo de que se pueda añadir a $r(x)$ el vector $e(x)$ con la operación módulo 2, y así obtener el mensaje original $c(x)$ [2].

- Calcular los síndromes.

  Si se evalúa $r(x)$ para $x = \alpha^i$, para $i = 1, 2, 3, \ldots, 2t$, donde $\alpha, \alpha^2, \alpha^3, \ldots, \alpha^{2t}$ son elementos del campo. Por lo tanto, se obtiene la ecuación $r(\alpha^i) = c(\alpha^i) + e(\alpha^i)$, en donde

$c(\alpha^i) = 0$, ya que los elementos de campo son raíces del polinomio generador $g(x)$, y del cual $c(x)$ es divisible. La ecuación 2.5 muestra que los síndromes S_i solo dependen del error y no de la palabra código $c(x)$ [23].

$$S_i = r(\alpha^i) = e(\alpha^i) \tag{2.5}$$

Si todos los síndromes tienen como valor 0, esto significaría que no se introdujo error, y se debe pasar la entrada directamente a la salida [8].

- Determinar el polinomio localizador de errores.

  Para este bloque será necesario hacer una revisión de diferentes arquitecturas las cuales serán presentadas en la sección 2.3. Sin embargo, se debe tener la idea de hallar el polinomio localizador de errores por medio del cálculo anterior de síndromes.

- Hallar las raíces del polinomio localizador de errores (del inglés error locator polynomial, nemónico ELP).

  Finalmente, se hallan las raíces del ELP evaluando para $x = 1, \alpha, \alpha^2, \ldots, \alpha^{n-1}$. Entonces, si α^i es una raíz cuando $\Lambda(\alpha^i) = 0$, además, ya que la palabra código recibida tiene una longitud n, α^{n-i} corresponderá al número de posición del error de $r(x)$. El proceso por el cual se encuentran las raíces del ELP es denominado búsqueda de Chien [5] [8].

En la figura 2.7, se presenta el diagrama de flujo que corresponde al decodificador, en el cual se muestran los cálculos necesarios anteriormente descritos para encontrar la palabra código trasmitida por el codificador.

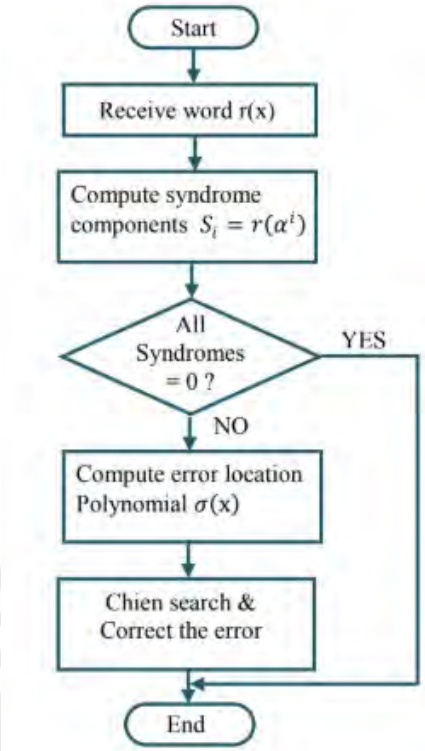

Figura 2.7: Diagrama de flujo del decodificador BCH [8]

2.3. Arquitecturas para implementación de decodificador BCH

Los códigos BCH operan en campos Galois, en donde se deben realizar operaciones que generan un mayor tiempo de cálculo. Estos pueden ser implementados en software o hardware; sin embargo, en las figuras 2.8 y 2.9 se corroboran las ventajas de una implementación en hardware respecto a una en software. En la figura 2.8, se tiene una comparación del área ocupada para software y hardware en función de la cantidad de errores corregidos. En la figura 2.9, se muestra un cuadro comparativo para el tiempo de cálculo en software y hardware del código BCH (15,5) tanto para un codificador como para un decodificador. Como se explicó en la sección 2.2, el código a implementarse será BCH (63,56) el cual tiene la capacidad de corregir 1 bit; por lo tanto, se tendría mayor ventaja en el área ocupada pues sería inferior a la de una implementación en software como se observa en la figura 2.8. Por otro lado, debido al procesamiento paralelo en hardware, el tiempo de cálculo es inferior al de software, además, la diferencia de estos tiempos se incrementa conforme el valor de n aumenta. En consecuencia, la implementación de los códigos BCH se realiza de manera eficaz en hardware [8].

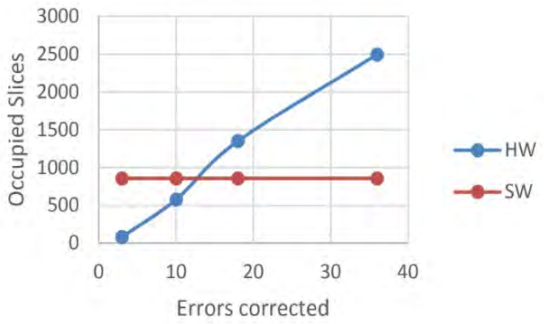

Figura 2.8: Área ocupada vs errores corregidos para software y hardware [9]

	$T_{hardware}$	$T_{software}$	n	k
Encoder	9.6106 µsec	1245.1 µsec	15	5
Decoder	40.22 µsec	13678.2 µsec		

Figura 2.9: Cuadro comparativo de tiempo de cálculo para BCH (n, k) [8]

La presente tesis tiene como propósito el diseño de una arquitectura de hardware para el decodificador BCH, por lo que es necesario conocer las diferentes arquitecturas que se tienen en la literatura y evaluarlas en referencia a parámetros como throughput, frecuencia, potencia y área. El decodificador se compone de 4 bloques, como se hizo referencia anteriormente; no obstante, no todos generan un retraso considerable o tiempo de cálculo elevado, como es el caso del bloque KES (del inglés Key Equation Solver, solucionador de ecuaciones clave) debido a su dificultad de implementación. Por ello se proponen diversas arquitecturas para el bloque KES.

Algunas de las arquitecturas a presentarse tienen como base el algoritmo Berlekamp-Massey (BM); sin embargo, en este se tienen desventajas que deben ser modificadas como la irregularidad de su arquitectura y el retraso de ruta que depende del número de errores que el código pueda corregir.

2.3.1. Arquitectura riBM

La arquitectura reformulada sin inversión Berlekamp-Massey (del inglés reformulated inversionless Berlekamp-Massey, nemónico riBM) fue propuesta en [24] con el objetivo de eliminar el cuello de botella debido al cálculo de discrepancias para códigos Reed-Solomon [23], [24]. Sin embargo, este puede ser utilizado en códigos BCH, además de ser simplificado debido a que estos son códigos binarios.

Con esta arquitectura se disminuye el retraso de la ruta crítica; no obstante, no se consiguen mejoras en la latencia y el número de componentes, es más, se incrementa el número de multiplicadores y multiplexores, y se mantiene el de sumadores.

2.3.2. Arquitectura RiBM

Esta arquitectura fue presentada como una reorganización de su precedente, riBM. El cambio fundamental es que la reorganización genera que los elementos de procesamiento (del inglés processing elements, nemónico PE) sean iguales con lo que se tiene una menor complejidad que en riBM [24], [25], [26]. Sin embargo, en esta arquitectura no se presenta una reducción en el uso de sumadores, multiplicadores y multiplexores de campos de Galois respecto a riBM, lo que significa una mayor área y por tanto mayor consumo de energía [25].

2.3.3. Arquitectura DcRiBM

La arquitectura RiBM sin cálculo de discrepancia (del inglés discrepancy computationless RiBM, nemónico DcRiBM) tiene como objetivo reducir la complejidad de hardware en sumadores y multiplicadores. Para ello, se elimina el bloque de control de cálculo de discrepancia, además de una reducción en el número de PE. No obstante, la cantidad de multiplexores se ve incrementada a más del doble, y por tanto el retraso de ruta crítica resulta mayor en el doble del periodo de los multiplexores al de RiBM [25]. Además de un ligero decremento en el troughput. Esto será expuesto en la tabla 2.3.

2.3.4. Arquitectura Peterson

Esta arquitectura es una de las más implementadas, debido a sus ventajas en la disminución de área de hardware, cuando el número de errores a corregir sea menor o igual a 4. Esto implica un menor consumo de energía, y mejoras en la velocidad de procesamiento. Además, esta arquitectura tiene un cálculo simple para encontrar los coeficientes del polinomio localizador de errores. En caso se requiera una capacidad de corrección de errores mayor, la complejidad y el área incrementan, y por tanto el consumo de energía se eleva [23], [8], [26].

2.3.5. Arquitectura I-Peterson

En la anterior arquitectura Peterson se tienen inversiones o divisiones que incrementan su complejidad. La arquitectura de Peterson sin inversión (del inglés, Inverssion-Less Peterson, abreviatura I-Peterson) elimina la operación de división mediante una multiplicación a los coeficientes, con ello se logra una reducción de área, consumo de energía y retraso de ruta crítico, además de un incremento de la velocidad de operación [26].

2.3.6. Análisis de arquitecturas

Como se presentó en la sección 2.2 el código que se implementará por recomendación de la CCSDS es el BCH (63,56), el cual posee una capacidad de corrección de error de 1 bit. Por lo tanto, el uso de una arquitectura iBM o alguno de sus derivados antes presentados (2.3.1, 2.3.2, 2.3.3) no serían recomendados, ya que se tendría un consumo de energía elevado por sus iteraciones, además de un retraso de ruta crítico elevado lo que incrementa el tiempo de decodificación. Una arquitectura Peterson o I-Peterson serían las adecuadas cuando se tenga como requerimiento una baja complejidad y consumo de energía. Para el caso de $t = 1$, no se tienen diferencias entre Peterson e I-Peterson, ya que no se cuenta con una división [8], [26]. En la tabla 2.3, se presenta la comparación de las arquitecturas expuestas en los parámetros latencia, retraso de ruta crítica y troughput.

Tabla 2.3: Comparación de diferentes arquitecturas

Arquitectura	Latencia	Retraso de ruta crítica	Throughput (GB/s)
riBM	$2t$	$T_{mul} + T_{add}$	-
RiBM	$2t$	$T_{mul} + T_{add}$	3 [3]
DcRiBM	$2t$	$T_{mul} + T_{add} + 2 \cdot T_{mux}$	2,9 [3]
Peterson	-	6.47 ns [2]	-
I-Peterson	$< 2t$	1.51 ns [2]	-

2.4. Modelo de solución del diseño

Debido al análisis en la sección 2.3, la arquitectura a implementarse para determinar el polinomio localizador de errores es Peterson. Como se presenta en la referencia [23], cuando el número de errores a corregir se encuentra en el rango de [1-4], el algoritmo de Peterson es eficiente, debido a que se pueden encontrar los coeficientes del polinomio por medio de reducciones y sin realizar un proceso iterativo como se hace en el algortimo de Berlekamp.

En la ecuación 2.6, se tiene el ELP, donde el coeficiente $\Lambda_0 = 1$.

$$\Lambda(x) = \prod_{l=1}^{v}(1 - \beta_l x) = \Lambda_0 + \Lambda_1 x + ... + \Lambda_v x^v \tag{2.6}$$

Los síndromes anteriormente calculados se relacionan con β_l como se muestra en la ecuación 2.7, y por tanto con ellos se puede encontrar Λ_v

[2] Implementación en el campo $GF(2^{14})$ [26].

[3] Implementado en el código BCH (2040,1930) [25].

$$S_i = \sum_{l=1}^{v} \beta_l^i \tag{2.7}$$

En tabla 2.4 se indican los coeficientes en función de los síndromes para diferentes t.

Tabla 2.4: Coeficientes de ELP en función de síndromes

Número de errores	Coeficientes del ELP
$t = 1$	$\Lambda_1 = S_1$
$t = 2$	$\Lambda_1 = S_1$ $\Lambda_2 = \frac{S_3 + S_1^3}{S_1}$
$t = 3$	$\Lambda_1 = S_1$ $\Lambda_2 = \frac{S_5 + S_1^2 S_3}{S_1^3 + S_3}$ $\Lambda_3 = (S_1^3 + S_3) + S_1 \Lambda_2$
$t = 4$	$\Lambda_1 = S_1$ $\Lambda_2 = \frac{S_1(S_7 + S_1^7) + S_3(S_1^5 + S_5)}{S_3(S_3 + S_1^3) + S_1(S_5 + S_1^5)}$ $\Lambda_3 = (S_1^3 + S_3) + S_1 \Lambda_2$ $\Lambda_4 = \frac{(S_5 + S_1^2 S_3) + (S_1^3 + S_3)\Lambda_2}{S_1}$

Conclusiones

De acuerdo a las referencias revisadas y el estudio realizado en los capítulos anteriores, se concluye que es necesaria la implementación de un decodificador BCH (63,56) que permita la corrección de 1 bit errado en una de las 63 posiciones de la palabra de entrada. Como se mostró en el capítulo 2, se tienen 3 bloques que permiten dicha decodificación, entonces el diseño propuesto, para una mayor eficiencia, se debe basar en una máquina de estados algorítmica con datapath (ASM-D) que sea capaz de realizar las funciones de los bloques siempre que sea necesario. Es decir, los bloques de la ASM-D debe ser como mínimo los siguientes:

- Bloque inicial en donde se necesita un requerimiento para inicializar los valores de registros.
- Bloque encargado de calcular el síndrome S y su respectivo peso de Hamming w.
- Bloque de decisión respecto al valor de w; es decir, este dará información de si es posible o no la decodificación de la palabra, si existe error en la palabra o no, o si el error se encuentra en los bits de paridad o información.
- Bloque de no decodificación, en caso la palabra posea de 2 a más errores; en este caso, será necesaria una señal que indique que no se pudo realizar la operación.
- Bloque en donde la palabra de entrada pase directamente a la salida, es decir, no se tenga error.
- Bloque de corrección de error cuando este se encuentra en los bits de paridad.
- Bloque de corrección de error cuando este se encuentra en los bits de información.
- Bloque de espera donde se necesite un nuevo requerimiento y se dé por reconocida la petición.

El diseño a proponer se debe basar en lo siguiente:

- Realizar las operaciones sobre campo de Galois que permitan obtener los síndromes, y a partir de ellos realizar la búsqueda del error.
- Una máquina de estados algorítmica con datapath que permita optimizar la decodificación de la palabra de entrada.

Recomendaciones y trabajo futuro

Se recomienda la implementación del decodificador BCH en hardware, debido a su menor tiempo de cálculo por procesamiento en paralelo, el cual es relevante en las comunicaciones satelitales. De acuerdo a las referencias revisadas, el hardware recomendado es un FPGA; sin embargo, se debe considerar la alta radiación en el espacio. Esto último hace referencia a que en la implementación del decodificador BCH para un nanosatélite se debe considerar un FPGA Antifusible por su baja sensibilidad a la radiación.

Para un trabajo futuro se debe considerar que el diseño de la arquitectura Peterson del decodificador puede ser implementado en Verilog y verificado con apoyo de Matlab. Es decir, en primer lugar se debe realizar un código en software (Matlab) que permita realizar las operaciones sobre Campos de Galois y así obtener los síndromes, además de la posición del error.
Será necesario realizar una síntesis comportamental la cual se basa en la conversión de un pseudocódigo a un nivel de transferencia de registros (RTL por sus siglas en inglés). Por otro lado, se debe considerar el diseño de la máquina de estados algorítmica con datapath (ASM-D), el cual debe constar únicamente de acciones y condiciones booleanas, mientras que el datapath se encargará de las interconexiones de componentes combinacionales y/o secuenciales.

Bibliografía

[1] Instituto de radioastronomía INRAS - PUCP, "Satélites: PUCP-Sat-1 - PUCP — Instituto de Radioastronomía."

[2] S. Arunkumar and T. Kalaivani, "FPGA implementation of CCSDS BCH (63, 56) for satellite communication," in *International Conference on Electronic Devices, Systems, and Applications*, pp. 248–253, 2012.

[3] B. Chen, "Hardware Implementation of Error Control Decoders," 2008.

[4] F. R. Lone, A. Puri, and S. Kumar, "Performance Comparison of Reed Solomon Code and BCH Code over Rayleigh Fading Channel," *International Journal of Computer Applications*, vol. 71, no. 20, pp. 23–26, 2013.

[5] M. Z. Hasan, M. A. Akbar, and I. Mahmood, "Performance Analysis of (63,56) Bch Code Using Multipath Rayleigh Fading Channel on Spartan-3 FPGA," vol. 2, p. 5, 2008.

[6] V. P. Mahadevaswamy, S. L. Sunitha, and B. N. Shobha, "Implementation of Fault Tolerant Method Using BCH Code on FPGA," no. 4, pp. 180–182, 2012.

[7] H. F. Abdulsada, "Design and Implementation of 2 bits BCH Error Correcting Codes using FPGA," no. October, 2017.

[8] S. JasamMohammed and H. Fadhil Abdulsada, "FPGA Implementation of 3 bits BCH Error Correcting Codes," *International Journal of Computer Applications*, vol. 71, no. 7, pp. 35–42, 2013.

[9] B. Jarvis, "FPGA Implementation and Analysis of Error Correction Codes for Physical Unclonable Functions," vol. 2, pp. 1–6.

[10] K. Hambleton, "Artemis I Map — NASA."

[11] K. F. Robinson, S. F. Spearing, and D. Hitt, "NASA's Space Launch System: Opportunities for Small Satellites to Deep Space Destinations," *32nd Annual Small Satellite*, no. August, pp. 1–9, 2019.

[12] Balta, "ERROR CORRECTION ALGORITHMS IN SATELLITE COMMUNICATION – BCH ENCODING / DECODING IMPLEMENTATION on VHDL,"

[13] M. Hatamian, H. Barati, S. Berenjian, A. Naghizadeh, and B. Razeghi, "Error Control Coding in Optical Fiber Communication Systems : An Overview," vol. 4, no. 2, pp. 70–80, 2015.

[14] R. Mehra, G. Saini, and S. Singh, "FPGA based high speed BCH encoder for wireless communication applications," *Proceedings - 2011 International Conference on Communication Systems and Network Technologies, CSNT 2011*, no. 2, pp. 576–579, 2011.

[15] Specialists Group on Coding for Visual Telephony, "Comparison Bose-Chaudhuri-Hocquenghem BCH and Reed Solomon,"

[16] C. Engineering, "VHDL IMPLEMENTATION OF REED-SOLOMON CODING,"

[17] CCSDS, "The Consultative Committee for Space Data Systems (CCSDS)."

[18] C. P. S, "Instroduction to Bose Chaudhuri Hocquenghem codes," 1967.

[19] L. Alejos, "Introducción a Tecnologías de Redes vía Satélite Cobertura geográfica," 2011.

[20] F. D. E. El and D. B. D. Carvalho, "Dois Códigos Corretores de Erro BCH para Comunicacoes Ópticas: Implementacao em FPGA e Comparacoes," 2015.

[21] Standford, "GF(2m) arithmetic: summary," tech. rep.

[22] Consultative Committee for Space Data Systems (CCSDS), "TC Synchronization and Channel Coding Recommendation for Space Data System Standards," *Ccsds 231.0-B-3*, no. 3, 2017.

[23] X. Zhang, *VLSI Architectures for Modern Error-Correcting Codes*. 2015.

[24] D. V. Sarwate and N. R. Shanbhag, "High-Speed Architectures for Reed – Solomon," vol. 9, no. 5, pp. 641–655, 2001.

[25] S. Yoon and H. Lee, "A discrepancy-computationless RiBM algorithm and its architecture for BCH decoders," *2008 IEEE International SOC Conference, SOCC*, pp. 379–382, 2008.

[26] S. An, H. Tang, and J. Park, “A inversion-less peterson algorithm based shared KES architecture for concatenated BCH decoder,” *ISOCC 2015 - International SoC Design Conference: SoC for Internet of Everything (IoE)*, pp. 281–282, 2016.