Universidad Nacional de Mar del Plata
Facultad de Ingeniería
Departamento de Ingeniería Electrónica

TRANSMISIÓN SEGURA EN COMUNICACIONES INLÁMBRICAS DE CORTO ALCANCE

Por
Ing. Leonardo José Arnone

Tesis presentada para optar por el Grado Académico de:
Doctor en Ingeniería, mención Electrónica
Director de Tesis: **Dr. Jorge Castiñeira Moreira**

Mar del Plata, Argentina. Mayo de 2008.

Universidad Nacional de Mar del Plata
Facultad de Ingeniería
Departamento de Ingeniería Electrónica

TRANSMISIÓN SEGURA EN COMUNICACIONES INLÁMBRICAS DE CORTO ALCANCE

Por
Ing. Leonardo José Arnone

Tesis presentada para optar por el Grado Académico de:
Doctor en Ingeniería, mención Electrónica
Director de Tesis: **Dr. Jorge Castiñeira Moreira**

Mar del Plata, Argentina. Mayo de 2008.

Índice General

Índice de figuras

Índice de tablas

Lista de símbolos

- α_{ij} constante auxiliar utilizada en el algoritmo de MacKay-Neal
- α_j constante auxiliar utilizada en el algoritmo de MacKay-Neal
- δQ_{ij} diferencia de los coeficientes Q_{ij}^a
- δR_{ij} diferencia de los coeficientes R_{ij}^a
- γ_b relación señal a ruido en un enlace infrarrojo
- γ_j costo de tener el bit y_j en 1, usado en PL
- λ utilizada en la implementación del decodificador LDPC por PL
- ρ separación entre el transmisor y el receptor en una transmisión infrarroja
- σ^2 varianza del ruido Gaussiano blanco y aditivo
- σ_m resultado que se tiene en el nodo paridad m cuando se evalúa en forma rígida la probabilidad a posteriori
- $\overline{\sigma}_m$ complemento de σ_m en módulo-2
- σ_n variable auxiliar usada en la sección 2.5.7
- a valor que puede tomar un símbolo binario $\{0, 1\}$
- $\mathbf{a}$ vector genérico usado en PL
- $\mathbf{A}$ submatriz de la matriz de paridad $\mathbf{H}$
- $\mathbf{A}$ matriz de coeficientes, usada en PL
- $a_{i,j}$ byte de la matriz de estado utilizada en el algoritmo AES
- $\mathbf{B}$ submatriz de $\mathbf{H}$

- $\mathbf{b}^T$ vector de restricciones, usado en PL
- b_i bits a la salida de un decodificador Hamming
- c factor de corrección constante
- $\mathbf{c}$ vector de código
- C código
- C_{ij}^a variable auxiliar utilizada en el algoritmo de MacKay-Neal
- $C_b(n,k)$ código de bloque de palabra de n bits, y k bits de información
- c_i bit de chequeo de paridad en un decodificador Hamming
- c_j coeficiente de coste, usado en PL
- C_{LDPC} codificador LDPC
- $C(x)$ polinomio usado en el algoritmo AES
- $\mathbf{d}$ vector símbolo
- $\hat{\mathbf{d}}$ es una estimación del vector $\mathbf{c}$
- $\hat{d}_j$ estimación del valor del símbolo en la posición j
- d_j nodo símbolo
- $d_{mín}$ distancia mínima
- $\mathbf{e}$ vector error
- E_b energía del pulso sin codificar
- E_c energía del pulso codificado
- e_i componente i-ésima de $\mathbf{e}$
- $e(X)$ polinomio error
- $\mathbf{f}$ solución optima, usada en PL
- $\mathbf{F}$ región factible, usada en PL
- f_j variable, usada en PL

- f_j^a probabilidad de que el j-ésimo símbolo sea a
- $f(x)$ transformada que tiene la propiedad $f(f(x)) = x$
- $f_+(|a|, |b|)$, $f_-(|a|, |b|)$ tablas de búsqueda para un decodificador LDPC con entradas $||a| - |b||$
- $\mathbf{G}$ matriz generadora del código
- GF campo de Galois
- $\mathbf{g_i}$ vector columna de la matriz generadora $\mathbf{G}$
- g_i coeficiente de $g(X)$
- $G_n(f)$ densidad espectral de potencia
- $g(t)$ pulso arbitrario
- $g(X)$ polinomio generador
- $g(x_i)$ factor de corrección
- H hiperplano, usado en PL
- $\mathbf{H}$ matriz de paridad
- $\mathbf{H}'$ matriz $\mathbf{H}$ a la cual se le aplica el método de eliminación de Gauss
- h_i nodo de paridad
- H_{ij} elemento de la matriz $\mathbf{H}$
- $\mathbf{h_j}$ vector columna de la matriz de paridad $\mathbf{H}$
- $Ho(\rho)$ ganancia óptica del enlace infrarrojo
- $h(x)$ factor de corrección
- $\mathbf{I}_k$ submatriz identidad de dimensión $k \times k$
- I_{LED} corriente en un LED
- $I_{promedio}$ corriente promedio en un LED
- k bits de longitud de la palabra de información

- K nivel de memoria en un codificador convolucional
- $k_{i,j}$ byte de la llave utilizada en el algoritmo AES
- K_{LED} eficiencia óptica de un LED
- L secuencia genérica de bits
- L tamaño del bloque de la llave usada en el algoritmo AES
- L_T longitud del permutador de datos aleatorio de un turbo código
- $L(U)$ relación de probabilidad $P(U = 0)/P(U = 1)$
- m entero positivo
- $\mathbf{m}$ vector de mensaje
- m_i coeficiente de $m(X)$
- $M(j)$ representa el conjunto de subíndices de todos los nodos paridas que interactúan con el nodo símbolo d_j
- $M(j)\backslash i$ indica la exclusión del nodo i del conjunto $M(j)$
- $m(X)$ polinomio del mensaje
- n bits de longitud de la palabra de código
- n entero positivo, usado en PL
- N entero positivo
- $N(i)$ representa el conjunto de subíndices de todos los nodos símbolos que interactúan con el nodo de paridad h_i
- $N(i)\backslash j$ indica la exclusión del nodo j de $N(i)$
- N_i número de unos de la fila i de la matriz $\mathbf{H}$, usado en PL
- N_0 densidad espectral de potencia de ruido
- n_r ruido del canal
- $\mathbf{n_r}$ vector de ruido del canal
- p entero positivo, usado en PL

- p probabilidad de error del canal
- P politopo, usado en PL
- $\mathbf{P}$ submatriz paridad
- $\mathbf{P}^T$ traspuesta de la submatriz paridad $\mathbf{P}$
- p_{be} probabilidad binaria de error del pulso
- p_{bec} probabilidad binaria de error del pulso codificado
- p_{be_AES} probabilidad binaria de error usando el algoritmo AES
- p_e probabilidad de error genérica
- p_i bits de paridad, en un código Hamming
- P_r potencia recibida en la superficie de un detector infrarrojo
- $p(y_r|x_i)$ probabilidad condicional de tener y_r dando x_i
- P_t potencia promedio transmitida por un transmisor infrarrojo
- $P(U = a)$ probabilidad que $U = a$
- p_{we} probabilidad de error por paquete de información
- $P(\mathbf{z}/\mathbf{u}^{(m)})$ probabilidad condicional que la secuencia $\mathbf{z}$ corresponda a una secuencia $\mathbf{u}^{(m)}$
- q entero positivo, usado en PL
- $\mathbf{Q}$ submatriz auxiliar
- Q_{ij}^a indica la probabilidad de que el nodo de paridad i se encuentre en el estado a
- Q_j^a estimación a posteriori
- $Q(x)$ función Q
- r velocidad sobre el canal
- $\mathbf{r}$ vector recibido con posibles errores
- R respuesta del detector infrarrojo

- r_b velocidad binaria
- R_c tasa del código
- R_{ij}^a informa la probabilidad que la paridad del nodo i se satisfaga, suponiendo que le nodo símbolo j se encuentra en el estado a
- $|r_n|$ variable auxiliar usada en la sección 2.5.7
- $r(X)$ polinomio de redundancias
- s número de unos por columna de la matriz $\mathbf{H}$
- $\mathbf{s}$ vector de síndrome
- S subespacio
- s' dimensión del espacio fila de la matriz $\mathbf{H}$
- s_{ij} resultado de la comparación entre $|wq_{ij}^0|$ y $|wq_{ij}^1|$
- $s_{mín}$ sensibilidad mínima de un fotodetector
- $s(X)$ polinomio síndrome
- t capacidad de corrección de errores
- T período de la palabra
- T_{AES} factor de propagación de error del algoritmo AES
- T_b período señal binaria
- $\mathbf{u}$ n-upla
- U variable aleatoria binaria
- $\mathbf{u}^{(i)}$ vector de salida i del codificador convolucional
- $u(X)$ polinomio del código
- $u^{(i)}(X)$ polinomio $u(X)$ desplazado i veces
- v número de unos por fila de la matriz $\mathbf{H}$
- V variable aleatoria binaria
- V_n espacio vectorial de todos los vectores de longitud n

- $|w\delta q_{ij}|$ diferencia de los coeficientes $|wq_{ij}^a|$
- $|w\delta r_{ij}|$ diferencia de los coeficientes $|wr_{ij}^a|$
- $|wc_{ij}^a|$, $|wc_j^a|$ variables auxiliares usados en el algoritmo de decodificación LDPC de suma-resta
- $|wf_j^a|$, $|wq_{ij}^a|$, $|wq_j^a|$, $|wr_{ij}^a|$ valor absoluto del log natural de f_j^a, Q_{ij}^a, Q_j^a, R_{ij}^a respectivamente
- $|wz|$ valor absoluto del log natural de z
- $\mathbf{x}$ palabra de información
- $\mathbf{x}$ solución factible, usada en PL
- $\hat{\mathbf{x}}$ estimación de la palabra de información original
- $\tilde{\mathbf{x}}$ solución optima del problema de PL
- x_i variable de decisión usada en PL
- x_i bits de información en un código Hamming
- $\hat{x}_n$ variable auxiliar usada en la sección 2.5.7
- $x_i(t)$ señal binaria
- $X(t)$ potencia óptica instantánea en una transmisión infrarroja
- $\mathbf{y}$ palabra de código
- $\mathbf{y}$ vector de variables de holgura, usada en PL
- y_j j-ésimo símbolo de la palabra de código $\mathbf{y}$
- $Y(t)$ corriente producida en un fotodetector
- $\mathbf{y_r}$ palabra de código recibida
- $y_r(t)$ señal binaria recibida
- z número real positivo, tal que $z \leq 1$
- Z función a maximizar o minimizar en un problema de PL
- z_i variable utilizada en el decodificador LDPC por PL

- $z(X)$ polinomio recibido
- z_n salida del decodificador usado en la sección 2.5.7

Apéndice A

- a,b,c log natural de A, B, C respectivamente
- A, B, C números reales, tal que $A \leq 1$, $B \leq 1$, $C \leq 1$

Apéndice C

- a, b, *selec* entradas tipo bit
- ax, bx *nosel* señales tipo bit
- *salida* salida tipo bit

Acrónimos

Acrónimos en castellano

- CCSR codificador convolucional sistemático recursivo.
- CD corrección en directa.
- CR corrección por retransmisión.
- LOG MPA logaritmo de la máxima probabilidad a posteriori.
- MP máxima probabilidad.
- MPA máxima probabilidad a posteriori.
- PL programación lineal.
- PM programación matemática.
- RGBA ruido Gaussiano blanco y aditivo.

Acrónimos en inglés

- AES *advanced encryption standard*, o algoritmo de encriptamiento simétrico estándar.
- ARQ *automatic repeat request*, o corrección por retransmisión automática.
- ASIC *application specific integrated circuits*, o circuitos integrados de aplicaciones específicas.
- AWGN *additive white Gaussian noise*, o ruido Gaussiano blanco y aditivo.
- BCJR *Baht, Cocke, Jelinek, Raviv algorithm*, o algoritmo de Baht, Cocke, Jelinek, Raviv.
- BER *bit error ratio*, o tasa de error binario.

- CAD *computer aided design*, o diseño asistido por computadora.
- DD *direct-detection*, detección directa.
- DES *data encryption standard*, o encriptamiento de datos estándar.
- EEPROM *electrically-erasable programmable read-only memory*, o memoria de sólo lectura programable y borrable eléctricamente.
- EPROM *erasable programmable read-only memory*, o memoria de sólo lectura programable y borrable.
- FEC *forward error correction*, o corrección en directa.
- FPGA *field programable gate array*, arreglo lógico programable en campo.
- GF *Galois field*, o campo de Galois.
- HDL *hardware description language*, o lenguaje de descripción circuital.
- IM *intensity-modulation*, modulación por intensidad.
- LAN *local area network*, o redes de área local.
- LDPC *low-density parity-check codes*, o códigos de paridad de baja densidad.
- LED *light emitting diode*, o diodo emisor de luz.
- LSI *large scale integration*, o integración a gran escala.
- LUT *look-up tables*, o tablas de búsqueda.
- MAP *maximun a posteriori probability*, o máxima probabilidad a posteriori.
- MSI *medium scale integration*, o integración de media escala.
- MOS *metal oxide semiconductor*, o semiconductor de óxido metálico.
- NETLIST *connectivity of an electronic design*, o conectividad de un diseño electrónico.
- OOK *on off keying*, o conmutación si-no.
- PAL *programmable array logic*, o lógica de arreglo programable.
- PLA *programmable logic array*, o arreglo lógico programable.

- PLD *programmable logic device*, o dispositivo lógico programable.
- PROM *programmable read only memory*, o memoria de solo lectura programable.
- RAM *random access memory*, o memoria de acceso aleatorio.
- ROM *read only memory*, o memoria de solo lectura.
- RSCC *recursive systematic convolutional code*, o codificador convolucional recursivo sistemático.
- SNR *signal to noise ratio*, o relación señal a ruido.
- SOVA *soft output Viterbi algorithm*, o algoritmo de Viterbi con decisión suave.
- SRAM *static random access memory*, o memoria estática de acceso aleatorio.
- VHDL *very high speed integrated circuit hardware description language*, o lenguaje de descripción de circuitos integrados de muy alta velocidad.
- VLSI *very large scale integration*, o integración a muy alta escala.
- WEP *wired equivalent privacy*, o privacidad equivalente en una red inalámbrica.

Abstract

En los sistemas de transmisión de datos inalámbricos modernos es de vital importancia la robustez con respecto al ruido ambiental, la privacidad, el tamaño, la portabilidad y el costo del enlace empleado.

En estos sistemas de comunicaciones de corto alcance, los códigos de control de error son una parte esencial. Dependiendo del tipo de aplicación requerida en la transmisión de datos, no siempre es necesario utilizar la técnica de control de error más poderosa o sofisticada.

Teniendo en cuenta estos factores, el objetivo principal de esta tesis es el estudio de implementaciones en dispositivos lógicos programables de sistemas de comunicaciones que utilizan diferentes estrategias de control de error, que permiten un alto grado de privacidad y una excelente performance en la tasa de error. En este sentido se entiende por transmisión segura aquella que tiene en cuenta estos dos aspecto de la comunicación.

Este estudio se orienta a definir los detalles de la implementación en sí, y a analizar la complejidad asociada, intentando reducir esta complejidad sin deterioro importante de la performance del sistema ideal o teórico correspondiente.

La investigación está centrada en la implementación de estos enlaces en *hardware* mediante lógica programable. Particularmente se analiza la implementación de tres tipos de sistemas de enlaces de corta distancia:

- Sistemas de control de error para enlaces de baja complejidad.
- Sistemas de control de error para enlaces de alta complejidad.
- Sistemas de control de error para enlaces de alta complejidad con alto grado de privacidad.

La contribución de esta tesis está dada no solo por el análisis y forma de implementar los diferentes sistemas de control de error, sino también porque se muestra como es posible implementar en lógica programable decodificadores de gran complejidad

algebraica, usando solamente operaciones de suma y resta en punto fijo, lo cual permite implementar enlaces con control de error de muy baja complejidad en términos del hardware empleado, con muy buena performance.

Agradecimientos

Agradezco a mi director de tesis Dr. Jorge Castiñeira Moreira por su inestimable apoyo, dedicación y capacidad de transmisión de conocimientos que permitieron la concreción de este trabajo.

Agradezco también a mis compañeros de laboratorio Carlos Gayoso, Claudio González, Miguel Rabini y Juan Carlos García por su amistad y por ser fuente de consulta y consejo.

A mi esposa por su amor y comprensión.

Y a mis Padres que me transmitieron el gusto por la lectura y el estudio.

Mar del Plata , Argentina. Mayo de 2008. Leonardo José Arnone

Capítulo 1

Objetivos de la tesis

1.1. Objetivos de la tesis

En los sistemas de transmisión de datos inalámbricos modernos es de vital importancia la robustez con respecto al ruido ambiental, la privacidad, el tamaño, la portabilidad y el costo del enlace empleado. Teniendo en cuenta este conjunto de factores, y debido a que los códigos de control de error forman parte sustancial de un sistema de comunicaciones de corto alcance, en esta tesis se realiza el estudio de implementaciones en dispositivos lógicos programables, de sistemas que utilizan diferentes estrategias de control de error.

El objetivo esencial de la tesis es el análisis de la implementación de diferentes técnicas de control de errores en transmisión segura para sistemas de comunicaciones inalámbricas de corto alcance. Se pondrá particular énfasis en las estrategias de simplificación de dichas implementaciones, asumiendo conocida la faz teórica que fundamenta las diferentes técnicas del control de error a través de la consulta bibliográfica de la teoría del control de error.

Dependiendo del tipo de aplicación requerida en la transmisión de datos, no siempre es necesario utilizar la técnica de control de error más poderosa o sofisticada. Por tal motivo, los enlaces han sido implementados usando diferentes estrategias de control de error, las cuales varían fundamentalmente en el grado de complejidad que presentan, y en la capacidad de control de error que poseen. Así, en ciertos casos, las aplicaciones requerirán de códigos de capacidad baja o mediana que estarán caracterizados por implementaciones de relativa sencillez. En otros casos harán uso de técnicas de control de error más poderosas y sofisticadas, derivando en implementaciones más complejas. La combinación con algoritmos conocidos de encriptamiento eleva esta complejidad de implementación, pero adiciona altos de niveles de privacidad sin deteriorar la performance del sistema frente al ruido.

En esta tesis se han reportado aportes realizados con referencia a tres aspectos de las comunicaciones inalámbricas de corta distancia:

- Sistemas de control de error para enlaces de baja complejidad.
- Sistemas de control de error para enlaces de alta complejidad.
- Sistemas de control de error para enlaces de alta complejidad con alto grado de privacidad.

1.2. Organización de la tesis y contribuciones originales

En le capítulo 2 se realiza una introducción a los fundamentos del control de error y encriptamiento. Se describen en forma sucinta los conceptos básicos de la detección y corrección de errores, los códigos de bloques, y la detección de errores por síndrome.

También se tratan los principios de los códigos cíclicos, los códigos convolucionales, los códigos de matriz de paridad de baja densidad, que inglés se denominan *Low-Density Parity-Check Codes* (LDPC), y los códigos turbo.

Se brinda una introducción a la programación lineal, y se demuestra como es posible utilizarla para decodificar códigos LDPC.

Para tratar el tema de privacidad se introduce brevemente el algoritmo de encriptamiento simétrico estándar, en inglés *Advanced Encryption Standard* (AES). También se brinda una introducción a los enlaces infrarrojos.

En el capítulo 3 se describen a los dispositivos lógicos programables y a los lenguajes de descripción de circuitos.

1.2.1. Aportes realizados con respectos a sistemas de control de error para enlaces de baja complejidad

Dependiendo del tipo de aplicación requerida en la transmisión de datos, no siempre es necesario utilizar la técnica de control de error más poderosa o sofisticada.

Por tal motivo, en el capítulo 4 se han implementados enlaces que utilizan diferentes estrategias de control de error, las cuales varían fundamentalmente en el grado de complejidad que presentan, y en la capacidad de control de error que poseen. En particular se describe un sistema de transmisión para uso médico basado en rayos infrarrojos, de un codificador cíclico, y de un codificador-decodificador Viterbi. Las contribuciones derivadas de esta implementación son:

- L. Arnone; C. Gayoso; C. González; A. Ivorra; D. Marín. Sistema de transmisión para uso médico basado en rayos infrarrojos. VII Workshop de Iberchip. Montevideo, Uruguay, pag. 15, Marzo 2001.
- L. Arnone; C. Gayoso; C. González; A. Ivorra; D. Marín. Aplicaciones de la microelectrónica en biotelemetría. XIII Congreso Argentino de Bioingeniería, II Jornadas de Ingeniería Clínica. SABI 2001. Tucumán, Argentina, pag. 147, Septiembre 2001.

En estos dos trabajos se detalla un sistema de transmisión para uso médico basado en rayos infrarrojos. Éste es un sistema de telemetría completo, el cual utiliza un

código Hamming para la detección y corrección de errores. Se presenta un sistema inalámbrico de muy buena performance, factible de ser modificado para diferentes usos, y al ser una implementación modular, permite fácilmente cambiar el sistema de control de error que utiliza.

- C. Gayoso; C. González; L. Arnone; J.C. García. Desarrollo de un sistema integral de biotelemetría: Monitorización remota de señales biológicas. VIII Workshop de Iberchip. Guadalajara, México, Abril 2002.

En este artículo se detalla un sistema de telemetría para el monitoreo de señales biomédicas, a diferencia de los trabajos anteriores cuenta con la implementación de la base de datos utilizada para el análisis de los datos adquiridos. El sistema se puede adaptar con facilidad para monitorear otros entornos.

- L. Arnone; C. Gayoso; C. González; J.C. García; J. Castiñeira Moreira. Diseño de un codificador cíclico programable para ser utilizado en comunicaciones infrarrojas interiores. VIII Congreso Argentino de Ciencias de la Computación. Buenos Aires, Argentina, pag. 30, Octubre 2002.

- L. Arnone; C. Gayoso; C. González; J.C. García; J. Castiñeira Moreira. Implementación en lógica programable de un codificador cíclico para enlaces infrarrojos. III Jornadas sobre Computación Reconfigurable y Aplicaciones. Madrid, España, pag. 283-290, Septiembre 2003.

En estos trabajos se detalla la implementación de un codificador-decodificador cíclico. Está pensado para ser utilizado en el sistema de telemetría infrarrojo completo presentado en los trabajos anteriores. Proporciona una interesante mejora en el funcionamiento del enlace infrarrojo. También la implementación permite fácilmente modificar el sistema para poder utilizar cualquier código (n, k) cíclico.

- L. Arnone; C. Gayoso; C. González; J.C. Garcia; J. Castiñeira Moreira. Diseño de un decodificador Viterbi para ser utilizado en enlaces infrarrojos. IV Workshop de Iberchip. La Habana, Cuba, pag. 43, Marzo 2003.

- L. Arnone; C. Gayoso; C. González; J.C. García; J. Castiñeira Moreira. Implementación en lógica programable de un codificador cíclico para enlaces infrarrojos. III Jornadas sobre Computación Reconfigurable y Aplicaciones. Madrid, España, pag. 283-290, Septiembre 2003.

- L. Arnone; C. Gayoso; C. González; J. Castiñeira Moreira. Logic programmable implemetation of convolutional coding for indoors infrared links. X RPIC, X Reunión de Trabajo en Procesamiento de la Información y Control. San Nicolás, Argentina, pag. 781-785, Octubre 2003.

Por último, de acuerdo con el objetivo de proporcionar una metodología de implementar enlaces inalámbricos de baja complejidad; en estos artículos se presenta un esquema de codificación convolucional que incluye un decodificador Viterbi. Al agregar este esquema de codificación se produce una notable mejora en la performance del enlace infrarrojo.

1.2.2. Aportes realizados con respectos a sistemas de control de error para enlaces de alta complejidad

Siguiendo con el objetivo, de lograr implementar en lógica programable enlaces inalámbricos con una performance de calidad de funcionamiento superior y teniendo en cuenta que los códigos de paridad de baja densidad (LDPC) tienen su funcionamiento cercano al límite de Shannon.

En el capítulo 5 se describen implementaciones de alta complejidad. Se propone una modificación al algoritmo original Mackay-Neal para la decodificación de códigos de matriz de paridad de baja densidad (LDPC), obteniendo entonces un algoritmo que solo utiliza operaciones de sumas, restas en punto fijo y búsquedas en tablas, sin necesidad de realizar productos ni cocientes, ni el uso de aritmética de punto flotante.

El algoritmo propuesto permite trabajar con códigos LDPC de cualquier tipo de matriz paridad (incluidas las generadas aleatoriamente, las generadas con una determinada regla de construcción, como las quasi-cíclicas, sean de tipo sistemático, o no sistemático), y se adapta en forma paramétrica cualquiera sea la tasa del código k/n. Esto le otorga una amplia versatilidad de aplicación y permite la utilización de los códigos LDPC más eficientes, que son los que utilizan matrices generadas aleatoriamente.

La forma propuesta para convertir las operaciones de suma, resta, cociente y multiplicación en punto flotante a operaciones de suma-resta en punto fijo, puede ser utilizada no solo para códigos LDPC, sino también por otro tipo de sistemas de decodificación, como por ejemplo los códigos turbo.

Utilizando el algoritmo propuesto se implementó un decodificador LDPC en lógica programable de muy baja complejidad en términos del *hardware* empleado, con muy buena performance.

También en este capítulo se muestra como utilizando principalmente comparaciones y tablas de búsqueda, es posible también implementar en forma sencilla un decodificador de códigos de control de error que utilice programación lineal

Como resultado de este capítulo se pueden listar los siguientes artículos derivados:

- J. Castiñeira Moreira; P. G. Farrell. Essential of Error-Control Coding. Pub.

John Wiley and Sons, England, 2006. Section: 8.7, A Logarithmic LDPC Decoder pag 302-306.

- L. Arnone; C. Gayoso; C. González; J. Castiñeira Moreira. Implementación de códigos de paridad de baja densidad en lógica programable usando sumas y restas. IV Jornadas sobre Computación Reconfigurable y Aplicaciones. Barcelona, España, pages 431-439, Septiembre 2004.

- L. Arnone; C. Gayoso; C. González; J. Castiñeira Moreira. Algoritmo de suma-resta para implementar códigos de paridad de baja densidad en lógica programable. XI Workshop de IBERCHIP. Salvador Bahia, Brazil, pages 127-129, Marzo 2005.

- L. Arnone; C. Gayoso; C. González; J. Castiñeira Moreira. A LDPC logarithmic decoder implementation. VIII International Symposium on Communications Theory and Applications. Ambleside,UK, pages 356-361, July 2005.

- L. Arnone; C. Gayoso; C. González; J. Castiñeira Moreira. Sum-subtract fixed point ldpc logarithmic decoder. XI RPIC, XI Reunión de Trabajo en Procesamiento de la Información y Control Río Cuarto, Argentina, pages 8-9, Setiembre 2005.

- L. Arnone; C. Gayoso; C. González; J. Castiñeira Moreira. Sum-subtract fixed point LDPC decoder. SPL06, II Southern Conference on Programmable Logic. Mar del Plata, Argentina, pag. 155-161, Marzo 2006.

- L. Arnone; C. Gayoso; C. González; J. Castiñeira Moreira. Sum-subtract fixed point ldpc decoder. Latin American Applied Research, vol 37, pag.17-20, 2007.

Es estos trabajos se describen en forma detallada como implementar un decodificador LDPC utilizando operaciones de sumas, restas en punto fijo y búsquedas en tablas, sin necesidad de realizar productos ni cocientes, ni el uso de aritmética de punto flotante. En éstas publicaciones están reflejados la mayoría de los tópicos detallados en el capítulo 5.

Como se detallo anteriormente, al tener los códigos LDPC funcionamiento cercano al límite de Shannon, y como el algoritmo presentado en estas publicaciones, permite trabajar con cualquier tipo de matriz paridad, esto posibilita la implementación de enlaces inalámbricos de una performance óptima.

- L. Arnone; C. González; C. Gayoso; J. Castiñeira Moreira. Implementación paramétrica en FPGA de un decodificador LDPC para cualquier tipo de matriz paridad y tasa de código. Aprobado para publicar en el XIV Workshop de IBERCHIP. Puebla, México, Febrero 2008

En este trabajo se realiza la implementación práctica del algoritmo propuesto anteriormente. Se hace un análisis detallado del *hardware* empleado en la implementación, incluyendo los elementos lógicos, registros y memorias. En este trabajo se demuestra que es factible construir prácticamente un decodificador LDPC en lógica programable, que trabaja con cualquier matriz de paridad **H** y se adapta en forma paramétrica cualquiera sea la tasa del código k/n.

- Scandurra; A. Dai Pra; L. Arnone; I. Pasoni; J. Castiñeira Moreira. A genetic algorithm based decoder for low density parity check codes. XI RPIC, XI Reunión de Trabajo en Procesamiento de la Información y Control Río Cuarto, Argentina, page 10, Setiembre 2005.

- Scandurra; A. Dai Pra; L. Arnone; I. Pasoni; J. Castiñeira Moreira. A genetic algo- rithm based decoder for low density parity check codes. Latin American Applied Research, vol 36, pag. 169-172, 2006.

En estas publicaciones se muestra como también es posible utilizar un algoritmo genético para decodificar códigos LDPC. Su performance es similar al algoritmo de suma-producto y al de suma-resta, aunque debido a su complejidad, sólo se pudo probar para códigos de tamaño reducido.

Si bien la tesis no trata sobre algoritmos genéticos, sirve para demostrar que el algoritmo de suma-resta propuesto no tiene limitaciones en el tamaño de la matriz de control de paridad **H**.

1.2.3. Aportes realizados con respectos a sistemas de control de error para enlaces de alta complejidad con alto grado de privacidad

Actualmente hay una creciente demanda de todo tipo de redes inalámbricas, ya sea desde el entornos corporativos y empresarial hasta el ámbito familiar. Si bien las ventajas de este tipo de enlaces son notables, como contrapartida se tienen grandes problemas de seguridad, debido a que las redes inalámbricas son vulnerables al ataque de cualquiera que conozca como interceptar el enlace.

Como solución se recurre a codificar la información a ser transmitida. Una de las técnicas de encriptamiento más conocidas y utilizadas es la que se conoce como el algoritmo de encriptamiento estándar avanzado, en inglés *Advanced Encryption Standard* (AES).

Esta solución provoca una fuerte propagación de errores, de manera que cuando se lo utiliza en una transmisión inalámbrica por ejemplo, y frente a la acción del

ruido, la performance de tasa de error, en inglés *Bit Error Rate* (BER) se deteriora fuertemente.

Por tal motivo,en el capítulo 6 se describe como combinando eficientes técnicas de control de errores con el algoritmo AES es posible obtener un sistema de comunicación con una gran capacidad de encriptamiento y una excelente performance en la tasa de error (BER). Sin duda que la combinación de las técnicas de control de error y encriptamiento constituye una disciplina de alto interés, no del todo desarrollada aún, que abre una ventana para futuros desarrollos.

Como resultado de este capítulo se pueden listar los siguientes artículos derivados:

- L. Arnone; C. Gayoso; C. González; J. Castiñeira Moreira; L. Liberatori. Analisys and FPGA implementation of combined error-control and encryption schemes. SPL07, III Southern Conference on Programmable Logic. Mar del Plata, Argentina, pag. 87-90, Marzo 2007.

- L. Arnone; C. Gayoso; C. González; J. Castiñeira Moreira; L. Liberatori. Uso de códigos de decodificación iterativa para mejorar la transmisión de mensajes cifrados con AES. XIII Workshop de IBERCHIP. Lima. Perú, pag. 245-248, Marzo 2007.

- L. Arnone; C. Gayoso; C. González; J. Castiñeira Moreira; M. Liberatori. Security and BER performance trade-of in wireless communication systems applications. XII RPIC, XII Reunión de Trabajo en Procesamiento de la Información y Control. Río Gallegos, Argentina, pag. 56, Octubre 2007.

En estas publicaciones se presentan esquemas que combinan el uso del algoritmo de encriptación AES con códigos muy eficientes para el control de errores, como los códigos LDPC y turbo.

En estos esquemas, el nivel de privacidad se puede mejorar notablemente usando procedimientos de permutación de datos de naturaleza pseudo-aleatoria en los codificadores y decodificadores.

Al combinar eficientes técnicas de control de errores con el algoritmo AES, es posible obtener un sistema de comunicación con una gran capacidad de encriptamiento y una excelente performance en la tasa de error (BER).

Por último ilustran cómo es posible implementar fácilmente un sistema en lógica programable usando un decodificador LDPC que utiliza sólo sumas y restas realizadas en punto fijo y un encriptador-desencriptador AES de bloques de 128 bits y una llave de 128 bits.

Finalmente en el capítulo 7 se plantean las conclusiones y las líneas de investigación futuras.

Capítulo 2

Fundamentos del control de error y encriptamiento

2.1. Conceptos básicos de la detección y corrección de errores

Cuando datos digitales son transmitidos en un canal que posee ruido, es importante tener un mecanismo que permita recuperarlos y limitar el número de errores que se producen. Normalmente una palabra consiste en una serie de unos y ceros llamados bits, que se codifican mediante el agregado de bits redundantes.

La palabra que llega al receptor, generalmente difiere de la enviada por el transmisor, debido a la presencia del ruido del canal. El receptor examina los bits que arribaron y realiza una decisión para determinar si la palabra de código es válida. Este proceso se lo llama decodificación.

2.1.1. Componentes de un sistema de codificación

La figura 2.1 muestra los componentes básicos de un sistema corrector de errores [2], donde una palabra de información $\mathbf{x}$ de k bits de longitud se codifica obteniendo una palabra de código $\mathbf{y} = codif(\mathbf{x})$ de longitud $n > k$. La palabra así codificada es enviada en un canal que presenta ruido, recibiendo en el otro extremo del canal una palabra alterada $\mathbf{y_r}$. Esta palabra alterada es entonces decodificada obteniendo la palabra recuperada $\hat{\mathbf{x}} = decod(\mathbf{y_r})$, la cual es una estimación de la palabra original x. Los componentes básicos de un sistema corrector de errores son:

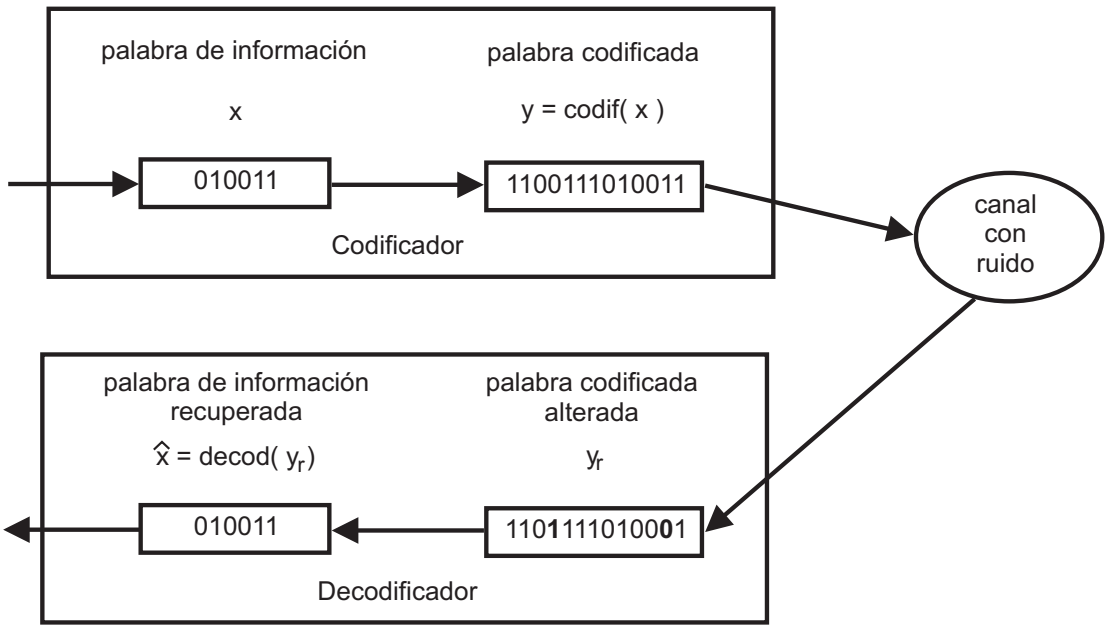

Figura 2.1: Componentes básicos de un sistema corrector de errores.

- **Palabra de información x:** es un bloque de símbolos que representan la información que se desea transmitir. La palabra de información se la denomina $\mathbf{x} \in \{0,1\}^k$, la cual tiene una longitud arbitraria de k bits. En esta tesis, se trabajan con bloques de códigos de longitud finita n. En caso de ser necesario enviar más de k bits de información, se transmiten múltiples bloques. En este trabajo se supone que cada bloque es independiente con respecto al contenido de información y el efecto del ruido.

- **Codificador:** se lo utiliza para agregar redundancia a la palabra de información. Básicamente es una función que convierte a la palabra de información $\mathbf{x}$ de k bits de longitud en una palabra binaria codificada $\mathbf{y}$ de n bits de longitud, con $n > k$.

- **Palabra codificada y:** es una palabra de n bits de longitud que se obtiene a la salida del codificador, y es la que se envía al canal de transmisión.

- **Código C:** es el conjunto de todas la palabras codificadas $\mathbf{y} \in C$ que pueden ser transmitidas.

- **Tasa del código R_c:** es la relación entre la longitud de la palabra de información y la palabra codificada, es decir $R_c = (k/n)$. Es deseable que el código tenga una tasa R_c elevada, así la información puede ser enviada más eficientemente.

- **Canal:** es el modelo del medio de comunicación en el cual se efectúa la transmisión. En este trabajo se considera que el canal está caracterizado por ruido Gaussiano blanco y aditivo (RGBA) que en inglés se conoce como *Additive White Gaussian Noise* (AWGN). El canal (RGBA) es uno de los modelos matemáticos más simples para varios canales de comunicación físicos, tal como las líneas de transmisión y algunos tipos de radio enlaces. Al ruido se lo representara como $\mathbf{n_r}$.

- **Palabra recibida $\mathbf{y_r}$:** es la palabra que se obtiene a la salida del canal en el receptor. Es la palabra codificada $\mathbf{y}$ alterada por el ruido del canal n_r, es decir $\mathbf{y_r} = \mathbf{y} + \mathbf{n_r}$.

- **Decodificador:** es el encargado de recuperar la palabra de información $\hat{\mathbf{x}}$ de la palabra recibida $\mathbf{y_r}$, donde $\hat{\mathbf{x}}$ es una estimación de la palabra de información $\mathbf{x}$ original.

2.1.2. Probabilidad de error en la transmisión de pulsos digitales

El efecto del ruido constituye uno de los principales problemas a resolver en el diseño de un sistema de comunicaciones. El otro problema presente en el diseño de un sistema de comunicaciones digitales es la interferencia intersimbólica [3, 4]. En esta sección se analiza el efecto del ruido como si la interferencia intersimbólica no existiera.

Si consideramos que la señal binaria está compuesta por los valores $x_1(t) = g(t)$ y $x_2(t) = -g(t)$ donde $g(t)$ es un pulso arbitrario de duración T_b. Dado que $x_1(t) = -x_2(t)$, esta señal es antipodal, y a la energía del pulso $g(t)$ se la llama E_b. Como indica [4], y se muestra en la figura 2.2 estas señales adoptan los valores $x_1 = \sqrt{E_b}$ y $x_2 = -\sqrt{E_b}$. Si suponemos que las dos señales son igualmente probables, y se

Figura 2.2: Señal binaria antipodal.

transmite la señal $x_1(t)$, entonces la señal recibida en el receptor es [4]:

$$y_r(t) = x_1(t) + n_r = \sqrt{E_b} + n_r \tag{2.1}$$

donde n_r representa el ruido Gaussiano blanco y aditivo (RGBA) con valor medio cero y varianza $\sigma^2 = N_0/2$, donde N_0 es la densidad espectral de potencia de ruido [3, 4]. Si se utiliza como umbral de detección el valor cero, se toma la decisión en favor de $x_1(t)$ si $y_r(t) > 0$, y en favor de $x_2(t)$ si $y_r(t) < 0$. Las dos funciones densidad de probabilidad condicionales son [4]:

$$p(y_r|x_1) = \frac{1}{\sqrt{\pi\, N_0}}\, e^{-(y_r - \sqrt{E_b})^2} \tag{2.2}$$

$$p(y_r|x_2) = \frac{1}{\sqrt{\pi\, N_0}}\, e^{-(y_r + \sqrt{E_b})^2} \tag{2.3}$$

Si la señal que se transmitió es $x_1(t)$, la probabilidad de error es simplemente la probabilidad de que $y_r(t) < 0$, es decir:

$$p(e|x_1) = \int_{-\infty}^{0} p(y_r|x_1)\, dy_r = Q\left(\sqrt{\frac{2\, E_b}{N_0}}\right) = Q\left(\frac{\sqrt{E_b}}{\sigma}\right) \tag{2.4}$$

donde $Q(x)$ es la función Q definida como:

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_k^{\infty} e^{-x^2/2}\, dx \tag{2.5}$$

En forma similar, si se transmite $x_2(t)$, se tiene $y_r(t) = -\sqrt{E_b} + n_r$ y la probabilidad de que $y_r(t) > 0$ es también:

$$p(e|x_2) = Q\left(\sqrt{\frac{2\,E_b}{N_0}}\right) \tag{2.6}$$

dado que las señales $x_1(t)$ y $x_2(t)$ tienen la misma probabilidad de ser transmitidas, la probabilidad promedio de error es:

$$p_{be} = \frac{1}{2}\,p(e|x_1) + \frac{1}{2}\,p(e|x_2) = Q\left(\sqrt{\frac{2\,E_b}{N_0}}\right) \tag{2.7}$$

al valor (E_b/N_0) se lo llama habitualmente la relación señal a ruido. Cuando se desea graficar la probabilidad de error p_{be} en función de la relación señal a ruido, es decir $p_{be} = f(E_b/N_0)$, es necesario expresar a (E_b/N_0) en decibeles, y es dependiente del valor de la amplitud de la señal binaria. En el caso de utilizar pulsos polares de amplitud ± 1, $(\pm\sqrt{E_b})$, la relación señal a ruido resulta ser igual a:

$$\left(\frac{E_b}{N_0}\right)_{dB} = 10 \cdot \log_{10}\left(\frac{E_b}{N_0}\right) = 10 \cdot \log_{10}\left(\frac{1}{N_0}\right) = 10 \cdot \log_{10}\left(\frac{1}{2\sigma^2}\right) \tag{2.8}$$

En el caso de utilizar una palabra codificada, con una tasa de código $R_c = (k/n)$, la energía del pulso es $E_c = R_c E_b = (k/n)E_b$ [5] y la probabilidad de error promedio p_{bec} resulta ser:

$$p_{bec} = Q\left(\sqrt{\frac{2\,E_c}{N_0}}\right) = Q\left(\sqrt{\frac{2\,R_c E_b}{N_0}}\right) \tag{2.9}$$

entonces ahora, la relación señal a ruido expresada en dB resulta ser igual a:

$$\left(\frac{E_b}{N_0}\right)_{dB} = 10 \cdot \log_{10}\left(\frac{1}{R_c N_0}\right) = 10 \cdot \log_{10}\left(\frac{1}{2R_c\sigma^2}\right) \tag{2.10}$$

2.2. Codificación y control de error

Una de las predicciones originadas por el segundo teorema de Shannon [6, 7] es que una suficientemente sofisticada técnica de codificación puede llevar a la transmisión sobre un canal ruidoso a operar como si se tratara de una transmisión sobre un canal libre de ruido. El teorema de Shannon demuestra que la transmisión puede ser libre de errores utilizando una técnica de codificación de naturaleza aleatoria.

En este proceso, las palabras de mensaje constituidas típicamente por bloques de bits, son asignadas de forma aleatoria a palabras de código de bits en una asignación o función biyectiva que produce redundancia, y que permite decodificar unívocamente cada mensaje, bajo ciertas condiciones. Esta codificación propuesta por Shannon es

esencialmente una codificación en bloques. Sin embargo, lo que no queda totalmente definido por el teorema es algún método constructivo para diseñar la sofisticada técnica de codificación.

Existen básicamente dos técnicas de codificación que difieren en la mecánica de la generación de redundancia. Estas dos técnicas básicas son la de la codificación en bloques, y la convolucional [8, 9]. En esta sección se analiza la codificación en bloques. En ésta, los errores podrán ser detectados o corregidos. En general, para un dado código, la capacidad de detección de errores es mas amplia que la capacidad de corregirlos, ya que para esto ultimo se necesita conocer la posición y magnitud del error.

2.2.1. Detección y corrección de errores

La codificación orientada a la detección del error es siempre más simple que la diseñada para corregirlos. El diseño de códigos especializados en detección o corrección depende de las características del sistema en donde se lo piensa aplicar. Cuando el sistema tiene la posibilidad de la comunicación duplex, es decir que existen los dos sentidos de la transmisión (como el caso de la línea telefónica) los códigos pueden ser diseñados solo para detectar los errores, ya que la corrección de los mismos se realiza por requerimiento de repetición.

Este esquema de transmisión se denomina de Corrección por Retransmisión (CR) que en Inglés se conoce como *Automatic Repeat reQuest* (ARQ). En todo sistema ARQ existe la posibilidad de requerir retransmisión. Cuando el sistema es de un solo sentido (como en el caso del *paging* o envío de mensajes por radiofrecuencia) no existe posibilidad de solicitud de retransmisión, por lo que la codificación empleada necesita corregir errores en el lado receptor. Esta forma de transmisión se denomina Corrección en Directa (CD) que en inglés se conoce como *Forward Error Correction* (FEC).

2.2.2. Códigos simples. El código de repetición

Una forma simple de realizar una codificación es sencillamente repetir el símbolo transmitido una determinada cantidad de veces. En la transmisión de símbolos binarios el bit '1' se representa por una secuencia de '1's , mientras el símbolo '0' se representa por una secuencia de '0's. Para el caso de un código de repetición por ejemplo, las palabras pertenecientes al código son (111) y (000). La primera típicamente representa al uno, mientras que la segunda representa al cero. Existirá detección de error cuando se reciba cualquier otra palabra de las ocho posibles que se tienen que no sean las del código. Así se sabrá que hay errores si se recibe por ejemplo la palabra (110).

Codificar significa básicamente ampliar la dimensión del sistema en que se trabaja para elegir en un espacio de vectores mayor, ciertas palabras como validas, mientras otras no pertenecen al código. En este caso se usan ocho vectores de los cuales solo dos pertenecen al código. Los otros seis patrones posibles de ser recibidos corresponden a patrones de error. Se dice entonces que el sistema es capaz de detectar uno o dos errores [9].

Considerando que la probabilidad de un error es mayor que la de tener dos errores, los patrones (110), (101) y (011) se considerarán secuencias de tres unos con un error. De acuerdo a este criterio, al recibir estas palabras el sistema corregirá el error asignándolo al hecho que un '1' fue el vector transmitido. De la misma forma, los patrones (001), (010) y (100) se considerarán secuencias de tres ceros que sufrieron un error. La corrección establecerá que el vector transmitido fue el de tres '0's por lo que el bit transmitido es el '0'. El sistema no podrá corregir dos errores.

Desde el punto de vista de la detección, el código no podrá darse cuenta de la presencia de tres errores, que hacen por ejemplo que una secuencia de tres ceros se convierta en otra de tres unos. Al recibir esta secuencia el sistema creerá que la palabra pertenece al código y lo recibirá erróneamente como un uno. Como se mencionó previamente, la eficiencia del código se mide como la relación entre los bits de información k y los bits de la palabra entera n que se transmite:

$$R_c = \frac{k}{n} \tag{2.11}$$

Los códigos de repetición tienen gran capacidad de corrección de errores, pero baja eficiencia de velocidad [9].

2.2.3. Códigos de bloques

Cuando se emplea codificación en bloques la información a codificar, esencialmente presentada en formato digital binario, es segmentada en bloques de k bits, que son los denominados bits de mensaje, que en conjunto constituyen 2^k posibles mensajes. El codificador transforma cada bloque de datos en un bloque normalmente más largo, de $n > k$ bits, denominados bits codificados o de la palabra código. En este esquema se observa que existen entonces $(n - k)$ bits que el codificador agrega a la palabra original, y que se denominan bits de redundancia o de control de paridad.

Como se explicó anteriormente, el proceso de codificación para el control de errores necesita de algún mecanismo de agregado de redundancia a la palabra original. Este proceso de redundancia se realiza de diferentes maneras operando sobre la información de mensaje, de forma que aplicando la operación inversa del lado receptor sea posible recuperar el mensaje original. En este sentido, en la operación final de

recepción o decodificación, los bits de redundancia son descartados, ya que no contienen información de mensaje. Este tipo de códigos se denomina código de bloques $C_b(n, k)$.

Asociada a estos códigos aparece lo que se denomina la tasa del código k/n, que mide el nivel de redundancia empleado en la codificación, significando el porcentaje de bits codificados que contienen información de mensaje. Este concepto tiene relación con el ancho de banda empleado en la transmisión cuando se utiliza codificación. Si por ejemplo se emplea una codificación en bloques de tasa $k/n = 2/3$ entonces se debe tener en cuenta que en el mismo tiempo T en que originalmente y sin codificar se alojan las dos señales que representan a los dos bits, se tiene ahora que poder alojar a tres señales con la misma ocupación temporal, de manera que el tiempo de duración de cada señal pasa a ser de $T/2$ a $T/3$, y la ocupación espectral es en consecuencia mayor.

Un concepto equivalente en el caso del almacenamiento de datos digitales es que la información codificada requerirá de mayor espacio físico para ser almacenada. Por estas razones será entonces conveniente mantener la tasa del código en niveles razonables, a pesar de que este objetivo va a estar en compromiso con la capacidad de corrección del código. Dado que los 2^k mensajes son transformados en palabras de n bits, se puede interpretar este procedimiento como la aplicación de una expansión del espacio vectorial de dimensión 2^k a otro de dimensión mayor 2^n, del cual se eligen de forma conveniente solo 2^k vectores.

2.2.4. Códigos de bloques lineales

La información a codificar se organiza en grupos de k bits que constituyen un vector de mensaje genérico $\mathbf{m} = (m_0, m_1, \ldots, m_{k-1})$, del cual existen 2^k posibles combinaciones. El codificador toma este mensaje y crea un vector de código $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1})$ de n componentes, donde normalmente $n > k$, es decir, se aplica redundancia.

Esto es básicamente una asignación biyectiva entre los 2^k vectores del espacio vectorial de mensaje y 2^k de los posibles 2^n vectores del espacio vectorial codificado. Cuando los valores de k y n son pequeños, la asignación puede realizarse a través de una tabla, pero cuando la magnitud de estas cantidades es grande, es necesario encontrar un método o mecanismo de generación del código. En este sentido la linealidad de las operaciones de este mecanismo simplifica grandemente el proceso de codificación. Se dice que un código de bloques de longitud n y 2^k palabras de mensaje es un código de bloques lineal $C_b(n, k)$, si las 2^k palabras de código forman un subespacio vectorial de dimensión k, del espacio vectorial V_n de todos los vectores de longitud n y componentes del campo $GF(2)$ [3, 5, 8, 9].

Codificar significa básicamente tomar las 2^k palabras binarias de k bits que se pretende codificar, y asignarlas a algunos de los 2^n vectores de n bits. Esto se realiza como una función unívoca entre los 2^k vectores y los 2^n vectores. Siendo regularmente $k < n$ existen más vectores de n bits que los que se tienen de k bits, con lo que la elección de los vectores de n bits debe hacerse empleando la menor redundancia, y maximizando la distancia o separación entre las palabras. En un código de bloques lineal, el conjunto de 2^k palabras constituye un subespacio vectorial del conjunto de vectores de n palabras. Como consecuencia de la definición, se puede establecer entonces que la suma de dos palabras cualesquiera del código será también otra palabra o vector del código.

2.2.5. Matriz generadora de un código de bloques G

Dado que un código lineal de bloques $C_b(n, k)$ es un subespacio vectorial del espacio vectorial V_n, será posible encontrar k vectores linealmente independientes que son a su vez palabras del código $\mathbf{g}_0, \mathbf{g}_1, \ldots, \mathbf{g}_{k-1}$. Estos vectores linealmente independientes se pueden organizar en la forma de una matriz, llamada matriz generadora:

$$\mathbf{G} = [\mathbf{g}_0, \mathbf{g}_1, \ldots, \mathbf{g}_{k-1}]^T \tag{2.12}$$

Si el vector de mensaje se expresa como $\mathbf{m} = (m_0, m_1, \ldots, m_{k-1})$, entonces la palabra o vector de código se obtiene como:

$$\mathbf{c} = \mathbf{m} \circ \mathbf{G} \tag{2.13}$$

De esta forma se establece un mecanismo matricial para la generación de las palabras del código.

2.2.6. Forma sistemática de un código de bloques

En esta forma, los vectores de código aparecen constituidos por $(n - k)$ bits de paridad seguidos por los k bits del vector mensaje. Esta forma de organizar la palabra codificada podría ser hecha al revés, es decir ubicando los bits de mensaje al principio de la palabra, y los de paridad al final. La manera en que esto sea hecho no modifica las propiedades de los códigos de bloques, aunque algunas expresiones matemáticas de las operaciones de codificación adoptan lógicamente una forma diferente en cada caso. Un código lineal sistemático de bloques (n, k) está especificado unívocamente por la matriz generadora de la forma:

$$\mathbf{G} = [\mathbf{P}\ \mathbf{I}_k] \tag{2.14}$$

Donde $\mathbf{I}_k$ es la matriz identidad de dimensión $k \times k$ y $\mathbf{P}$ es la matriz paridad de dimensión $k \times (n-k)$.

La matriz generadora $\mathbf{G}$ contiene k vectores fila linealmente independientes, que generan el subespacio vectorial S que pertenece al espacio vectorial V_n (ver sección 2.2.4). También V_n tiene a su vez asociado un subespacio vectorial dual S_d, que es generado por las filas de una matriz $\mathbf{H}$. A esta matriz $\mathbf{H}$ se la define como:

$$\mathbf{H} = [\mathbf{I}_{n-k}\ \mathbf{P}^T] \tag{2.15}$$

donde $\mathbf{P}^T$ es la traspuesta de la submatriz de paridad $\mathbf{P}$. Cada vector del espacio fila de la matriz $\mathbf{G}$ es ortogonal a las filas de la matriz $\mathbf{H}$ y viceversa.

La matriz $\mathbf{H}$ está construida de manera que el producto interno entre un vector fila $\mathbf{g_i}$ de $\mathbf{G}$ y un vector fila $\mathbf{h_j}$ de $\mathbf{H}$ sean ortogonales, es decir $\mathbf{g_i} \circ \mathbf{h_j} = \mathbf{0}$. Esto tiene como consecuencia que:

$$\mathbf{G} \circ \mathbf{H}^T = \mathbf{0} \tag{2.16}$$

Usando el resultado de la ecuación 2.13 surge entonces que:

$$\mathbf{c} \circ \mathbf{H}^T = \mathbf{m} \circ \mathbf{G} \circ \mathbf{H}^T = \mathbf{0} \tag{2.17}$$

2.2.7. Detección de errores por síndrome

Para un código lineal de bloques $C_b(n,k)$ se han definido las correspondientes matrices de generación $\mathbf{G}$ y paridad $\mathbf{H}$ y se ha visto que estando definido sobre el campo binario $GF(2)$, el vector de código es de la forma $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1})$, donde los $c_i \in GF(2)$. Como consecuencia de la transmisión de este vector a través del canal se produce el efecto del ruido y los elementos del vector son decodificados con posibles errores.

Se denomina $\mathbf{r} = (r_0, r_1, \ldots, r_{n-1})$ al vector recibido con posibles errores de forma, donde también los $r_i \in GF(2)$. Aparece entonces el vector error $\mathbf{e} = (e_0, e_1, \ldots, e_{n-1})$ cuyas componentes pertenecen al campo binario, $e_i \in GF(2)$, y que esta relacionado con los vectores anteriores por la ecuación:

$$\mathbf{e} = \mathbf{r} \oplus \mathbf{c} \tag{2.18}$$

Donde la operación $\oplus$ indica la adición en el campo binario $GF(2)$. El vector error tiene componentes distintas de cero solo en las posiciones donde se produjo algún error. Una vez conocido el patrón de error, se puede, en función del vector recibido y de éste vector de error, corregir la recepción y determinar una estimación valedera para la palabra recibida:

$$\mathbf{c} = \mathbf{r} \oplus \mathbf{e} \tag{2.19}$$

Dado que toda palabra o vector de código debe cumplir con la condición:

$$\mathbf{c} \circ \mathbf{H}^T = \mathbf{0} \tag{2.20}$$

se puede implementar un mecanismo de detección de error basado en la operación:

$$\mathbf{s} = \mathbf{r} \circ \mathbf{H}^T \tag{2.21}$$

donde el vector $\mathbf{s}$ se denomina síndrome. La operación de detección se basa en el hecho que operando sobre el vector recibido, el vector síndrome debiera ser un vector nulo para que tal vector recibido sea integrante del código. Dado que $\mathbf{r} = \mathbf{c} \oplus \mathbf{e}$, entonces:

$$\begin{aligned} \mathbf{s} &= \mathbf{r} \circ \mathbf{H}^T \\ &= (\mathbf{c} \oplus \mathbf{e}) \circ \mathbf{H}^T \\ &= \mathbf{c} \circ \mathbf{H}^T \oplus \mathbf{e} \circ \mathbf{H}^T \\ &= \mathbf{e} \circ \mathbf{H}^T \end{aligned} \tag{2.22}$$

Si el patrón de error es el vector nulo, el vector síndrome también será el vector nulo. Cuando este vector sea distinto del vector nulo, se estará en presencia de un vector recibido con errores. Existe sin embargo la posibilidad de que el vector síndrome sea el vector nulo aun cuando haya habido errores. En efecto si el número y posición de errores hace que un vector del código se transforme en otro vector del mismo código esta operación no estará en condiciones de detectar ese problema. En este caso se esta en presencia de un patrón de error no detectable, y por lo tanto fuera de la capacidad del código para operar convenientemente.

Los patrones de error no detectables son aquellos que cumplen con la condición $\mathbf{s} = \mathbf{e} \circ \mathbf{H}^T = \mathbf{0}$, es decir, aquellos patrones de error cuya forma coincide con vectores de código (Es decir $\mathbf{e} = \mathbf{c}$). Existen por lo tanto $2^k - 1$ patrones de error no nulos que no son detectables.

2.2.8. Códigos de Hamming

Una clase de códigos de bloques ampliamente utilizada es la de los códigos de Hamming [10]. Para cualquier entero positivo $m \geq 3$ existe un código de Hamming con las siguientes características:

- Longitud $n = 2^m - 1$
- N° de bits de mensaje $k = 2^m - m - 1$
- N° de bits de control de paridad $m = n - k$

- Capacidad de corrección de errores $t = 1(d_{mín} = 3)$ [5, 9]

La matriz de paridad de estos códigos $\mathbf{H}$ se forma con las columnas de m bits no nulas que puede ser implementada en la forma sistemática:

$$\mathbf{H} = [\mathbf{I}_m \ \mathbf{Q}] \tag{2.23}$$

donde la submatriz identidad es de $m \times m$ y la submatriz $\mathbf{Q}$ consiste de $2^m - m - 1$ columnas formadas con vectores de peso 2 [9] o mayor. La matriz generadora se obtiene de acuerdo a la siguiente expresión, para la forma sistemática del código:

$$\mathbf{G} = [\mathbf{Q}^T \ \mathbf{I}_{2^m-1}] \tag{2.24}$$

En la matriz $\mathbf{H}$ la suma de tres columnas dan como resultado el vector nulo, con lo cual la distancia mínima de estos códigos es $d_{mín} = 3$ [9] de forma que pueden ser utilizados para corregir patrones de error de un bit o detectar cualquier patrón de error de dos bits.

2.2.9. Filosofías de corrección en un sistemas de corrección de error FEC

Como se ha dicho los sistemas que emplean la filosofía de corrección FEC tienen la limitación de no poder contar con requerimiento de repetición, por lo que toda la potencia del código usado se emplea para la corrección de error [9]. La fuente de información genera una señal binaria de símbolos equiprobables que tiene una velocidad de transmisión r_b, de forma que el codificador toma tal información agrupada en k bits, para agregarle $n - k$ bits de corrección, constituyendo un código de bloque $C_b(n, k)$ cuya relación de eficiencia es $R_c = k/n$, siendo $R_c < 1$. La velocidad sobre el canal debe ser mayor que la velocidad de señalización r_b:

$$r = \left(\frac{n}{k}\right) r_b = \frac{r_b}{R_c} \tag{2.25}$$

El código utilizado tiene una distancia mínima $d_{mín} = 2t + 1$ y se analiza la respuesta del sistema frente a ruido blanco y Gaussiano con un canal con una probabilidad de error $p \ll 1$. La información de fuente tiene una energía de bit promedio E_b, de forma que la energía promedio por bit de código es $R_c E_b$. Entonces la relación de energía a densidad de ruido es:

$$\left(\frac{E_c}{N_0}\right) = \frac{R_c E_b}{N_0} = R_c \left(\frac{E_b}{N_0}\right) \tag{2.26}$$

El cociente entre la energía por bit E_b y la densidad espectral de ruido N_0 juega un papel muy importante en la caracterización de los sistemas de comunicaciones, y

será utilizado como parámetro de comparación. En este caso se diferencia su valor entre la situación codificada y la no codificada. Se calcula la probabilidad de error de los bits de mensaje, que se denomina p_{be}, para distinguirla de la probabilidad de error por paquete de información p_{we}.

Si el código corrige hasta t errores por paquete de información y suponiendo que la probabilidad de error del canal p es pequeña puede hacerse la aproximación [9]:

$$p_{we} \simeq \binom{n}{t+1} p^{t+1} \tag{2.27}$$

que básicamente significa que una paquete de información no corregido tiene $t+1$ errores. Si se transmiten Nk bits de información de fuente, siendo $N \gg 1$, la probabilidad de error es igual a:

$$p_{be} = \frac{1}{n}(t+1)p_{we} \tag{2.28}$$

La cual se puede aproximar como [9]:

$$p_{be} = \binom{n}{t-1} p^{t+1} \tag{2.29}$$

Cuando el sistema opera en presencia de ruido blanco y Gaussiano, de densidad espectral de potencia $G_n(f) = N_0/2$, y esta diseñado de forma que se ha empleado filtro adaptado y señalización polar, la probabilidad de error p_e esta dada por [9]:

$$p_e = Q\left(\sqrt{2\frac{E_b}{N_0}}\right) \tag{2.30}$$

luego la probabilidad de error del canal p es:

$$p = Q\left(\sqrt{2\frac{E_c}{N_0}}\right) = Q\left(\sqrt{2R_c\frac{E_b}{N_0}}\right) \tag{2.31}$$

La probabilidad binaria de error de un sistema FEC es entonces:

$$p_{be} = \binom{n-1}{t}\left[Q\left(\sqrt{2R_c\frac{E_b}{N_0}}\right)\right]^{t+1} \tag{2.32}$$

Un sistema no codificado tendrá una probabilidad de error:

$$p_{be} \simeq Q\left(\sqrt{2\frac{E_b}{N_0}}\right) \tag{2.33}$$

De la comparación del resultado de las expresiones 2.32 y 2.33 se puede establecer si existe realmente una mejora en el sistema por aplicación de la técnica de codificación para el control de errores. Esta comparación depende de los valores de t y R_c que la técnica de codificación ofrece. Cuando el sistema trabaja con un nivel de ruido grande puede suceder que la respuesta del mismo frente al ruido sea peor que la del sistema sin codificar.

2.3. Códigos cíclicos

Los códigos cíclicos binarios son fácilmente implementados mediante registros de desplazamiento realimentados. El cálculo del síndrome [4] también se realiza de forma muy sencilla mediante registros de desplazamiento.

Un código (n, k) se dice que es un código cíclico si cumple la siguiente propiedad [3, 8]: si una n-upla $\mathbf{u} = (u_0, u_1, u_2, ..., u_{n-1})$ es un vector del código en el subespacio S, entonces $\mathbf{u}(1) = (u_{n-1}, u_0, u_1, ...u_{n-2})$ obtenido mediante un desplazamiento; también es un vector del código en el subespacio S.

Los componentes del vector del código $\mathbf{u} = (u_0, u_1, u_2, ..., u_{n-1})$ pueden ser tratados como los coeficientes de un polinomio $u(X) = u_0 + u_1X + u_2X^2 + ... + u_{n-1}X^{n-1}$ donde la presencia o ausencia de cada término en el polinomio indica la presencia de un 1 o 0 en la correspondiente localización de la n-upla. En este caso un desplazamiento i es indicado como $u^{(i)}(X)$ y se obtiene mediante la siguiente operación:

$$u^{(i)}(X) = X^i\, u(X)\, mod(X^n + 1) \tag{2.34}$$

donde *mod* es la operación *módulo*. En un código cíclico (n, k), cada polinomio de la palabra de código $u(X)$ en el subespacio S puede ser expresado como $u(X) = m(x) \cdot g(X)$ donde $m(X)$ es el polinomio del mensaje y se escribe como:

$$m(X) = m_0 + m_1X + m_2X^2 + ... + m_{k-1}X^{k-1} \tag{2.35}$$

y $g(X)$ es el polinomio generador, y tiene la forma:

$$g(X) = g_0 + g_1X + g_2X^2 + ... + g_{n-k}X^{n-k} \tag{2.36}$$

donde g_0 y g_{n-k} deben ser igual a 1.

A continuación se describe una forma sistemática para obtener a partir del vector de mensaje $\mathbf{m} = (m_0, m_1, ..., m_{k-1})$ el vector del código $\mathbf{u} = (r_0, r_1, \ldots, r_{n-k-1}, m_0, m_1, \ldots, m_{k-1})$, donde al principio del vector se encuentran los $(n - k)$ bits de paridad y al final los k bits del mensaje. Al vector $\mathbf{u}$ le corresponde el polinomio de código $u(X)$ que entonces puede ser escrito como [3]:

$$\begin{aligned} u(X) &= r_0 + r_1X + \ldots + r_{n-k-1}X^{n-k-1} + \\ & m_0X^{n-k} + m_1X^{n-k+1} + \ldots + m_{k-1}X^{n-1} \end{aligned} \tag{2.37}$$

la ecuación 2.37 es equivalente a poner [3]:

$$u(X) = r(X) + X^{n-k}m(X) \tag{2.38}$$

de la ecuación 2.37 y 2.38 se ve que la codificación en forma sistemática usando códigos cíclicos se logra colocando los dígitos del mensaje en las k etapas más hacia

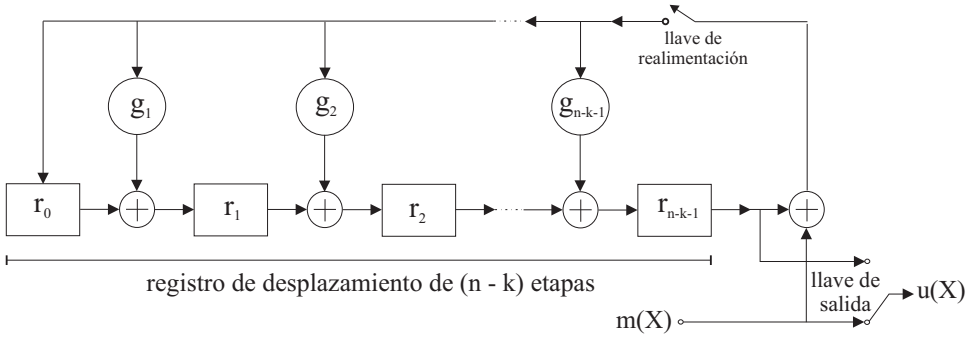

Figura 2.3: Codificador cíclico de $(n-k)$ etapas.

la derecha del registro de palabra de código y ubicando los dígitos de paridad en las $(n-k)$ etapas más hacia la izquierda. El polinomio $r(X)$ se obtiene de realizar la siguiente operación [3]:

$$r(X) = X^{n-k}\, m(X)\, mod(g(X)) \tag{2.39}$$

En la figura 2.3 se ilustra un circuito [3] que permite codificar un código cíclico en forma sistemática. El cálculo de los bits de paridad es el resultado de obtener $X^{n-k}\, m(X)\ \ mod(g(X))$, en otras palabras, es la división del polinomio de mensaje desplazado hacia la derecha por el polinomio generador $g(X)$. La codificación comienza inicializando con ceros los registros r_0 a r_{n-k-1}, cerrando la llave de realimentación, y colocando la llave de salida en la posición que permite que los bits del mensaje $m(X)$ pasen a la salida. Los k bits del mensaje son desplazados dentro del registro r y simultáneamente liberados hacia la salida. Luego de desplazar k veces, los registros r_0 a r_{n-k-1} contienen los bits de paridad. Entonces se abre la llave de realimentación, y la llave de salida se coloca en la posición que permite que los bits de paridad lleguen a la salida.

El vector de código transmitido, puede estar perturbado por ruido, por lo tanto, el vector recibido puede no coincidir con el transmitido. Si se supone que se transmite la palabra de código $u(X)$ representada por:

$$u(X) = m(X)\, g(X) \tag{2.40}$$

El polinomio $z(X)$ recibido estará formado por:

$$z(X) = u(X) + e(X) \tag{2.41}$$

donde $e(X)$ representa el polinomio de error. El síndrome $s(X)$ [4] es el resto de dividir $z(X)$ por $g(X)$, es decir:

$$z(X) = q(X)\, g(X) + s(X) \tag{2.42}$$

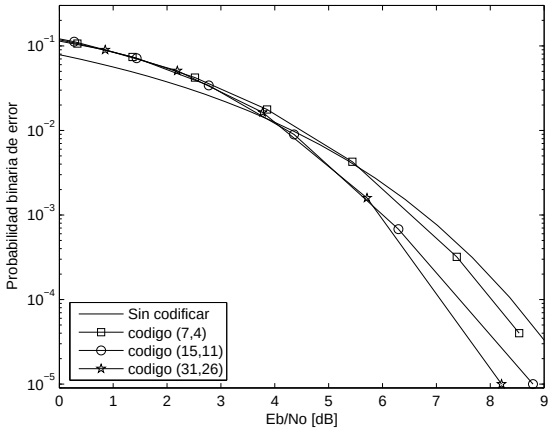

Figura 2.4: p_{be} para diferentes códigos cíclicos.

combinando las ecuaciones (2.40) a (2.42) se obtiene:

$$e(X) = [m(X) + q(X)]\, g(X) + s(X) \tag{2.43}$$

Observando las ecuaciones 2.42 y 2.43 se ve que el síndrome $s(X)$ obtenido como el resto de dividir $z(X)$ por $g(X)$ es el mismo polinomio que se obtiene del resto de dividir $e(X)$ por $g(X)$. Así el síndrome de la palabra de código recibida $z(X)$ contiene la información necesaria para la corrección de los errores producidos. El cálculo del síndrome se realiza mediante un circuito similar al circuito utilizado en el codificador.

2.3.1. Probabilidad de errores

A continuación se analiza qué ocurre con los errores producidos durante la transmisión cuando se utilizan códigos cíclicos. Se supone que el enlace es un canal simétrico binario con probabilidad de error por bit p, es decir, es igualmente probable que cuando se transmita un '0' se reciba un '1' ó cuando se transmita un '1' se reciba un '0'.

Suponiendo que la información es codificada en un código $[n, k, 2t+1]$, con $n-k = 2t+1$, se tiene que t es el número de errores que el código puede corregir. Entonces cuando ocurren $t+1$ errores, se produce una decodificación incorrecta. Se denomina p_{be} a la probabilidad de decodificar incorrectamente, cuando el número de errores excede a t. Se puede aproximar a p_{be} como [11, 12]:

$$p_{be} \leq \binom{n}{t+1} p^{t+1} (1-p)^{n-t-1} \tag{2.44}$$

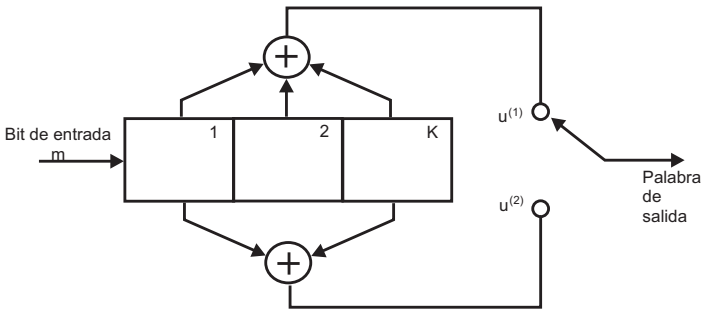

Figura 2.5: Codificador convolucional $(2, 1, 3)$.

Para poder comparar diferentes tipos de códigos se utiliza [3]:

$$p = Q\left(\sqrt{\frac{k}{n} \cdot \frac{E_b}{N_0}}\right) \tag{2.45}$$

En la figura 2.4 se grafica la probabilidad de error p_{be} para diferentes códigos cíclicos, en función del parámetro E_b/N_0, en dB. Se puede observar cómo se produce una disminución apreciable en los errores al codificar el mensaje.

2.4. Códigos convolucionales

Existen dos tipos de codificación para el control de error en un canal, los códigos de bloques y los códigos convolucionales [4, 8]. En un código convolucional los n bits de la salida del codificador dependen no sólo de los k bits de entrada, sino también de K bloques de información de entrada previos. Un código convolucional (n, k, K) se implementa con k entradas que ingresan a un circuito secuencial de n salidas con un nivel de memoria K. Típicamente n y k son números enteros siendo normalmente $k < n$. El nivel de memoria K debe hacerse grande para aumentar la capacidad de detección de error del código.

La figura 2.5 representa un codificador para generar un código convolucional binario $(2, 1, 3)$. El vector de entrada $\mathbf{m}$ es una secuencia que tiene la forma:

$$\mathbf{m} = (m_0,\, m_1,\, m_2,\, \cdots) \tag{2.46}$$

El vector de salida $\mathbf{u}$ se configura con las secuencias de salida $\mathbf{u}^{(1)}$ y $\mathbf{u}^{(2)}$:

$$\begin{aligned} \mathbf{u}^{(1)} &= (u_0^{(1)},\, u_1^{(1)},\, u_2^{(1)},\, \cdots) \\ \mathbf{u}^{(2)} &= (u_0^{(2)},\, u_1^{(2)},\, u_2^{(2)},\, \cdots) \end{aligned} \tag{2.47}$$

Un esquema general, de un codificador convolucional, emplea kK registros que operan en el campo de Galois $GF(2)$ (suma y multiplicación binaria en modulo-2).

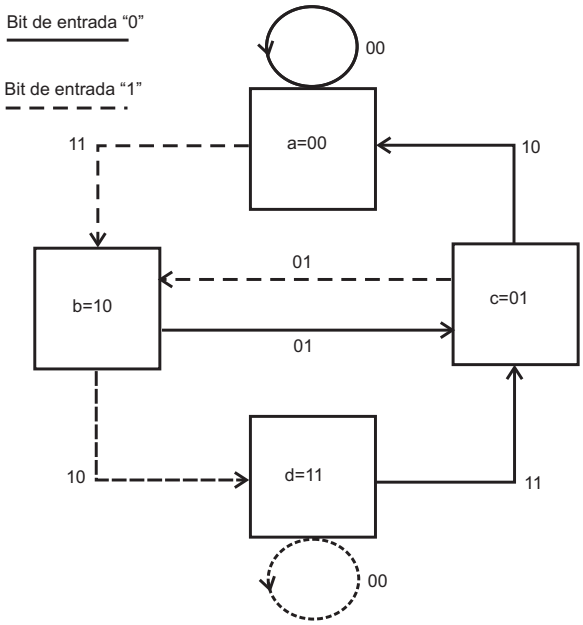

Figura 2.6: Diagrama de estados del codificador convolucional.

El parámetro K se denomina longitud de restricción, básicamente es una medida de la forma en que los bits previos afectan a los que se están generando en la salida.

En la figura se ve que las secuencias de entrada de k bits son desplazadas hacia la derecha en las primeras K etapas de los registros. La salida es muestreada para generar la secuencia de salida del código. Existen n bits de salida por cada k bits de entrada, por lo que se puede definir la velocidad del código como k/n siendo $k < n$. La mayoría de los códigos convolucionales emplean $k = 1$, de forma que los bits de entrada son ingresados de forma serie. Para este tipo de codificador convolucional la velocidad de código es $1/n$. El registro del sistema que tiene kK etapas, tendrá en este caso entonces K etapas.

2.4.1. Representación por diagrama de estados de un codificador convolucional

El estado de un código convolucional $1/n$ se define como el contenido de los $K-1$ registros de estado que se encuentran a la derecha. De esta manera los $K-1$ registros de estado de la izquierda constituyen el próximo estado. El diagrama de estados [3, 13] es una forma de representación de la evolución de las secuencias de estado para estos códigos. El codificador analizado en este ejemplo tiene un diagrama de estados como el que se ve en la figura 2.6. El sistema se analiza considerando que existen cuatro estados representados por $a = 00$, $b = 10$, $c = 01$ y $d = 11$. Existen sólo dos

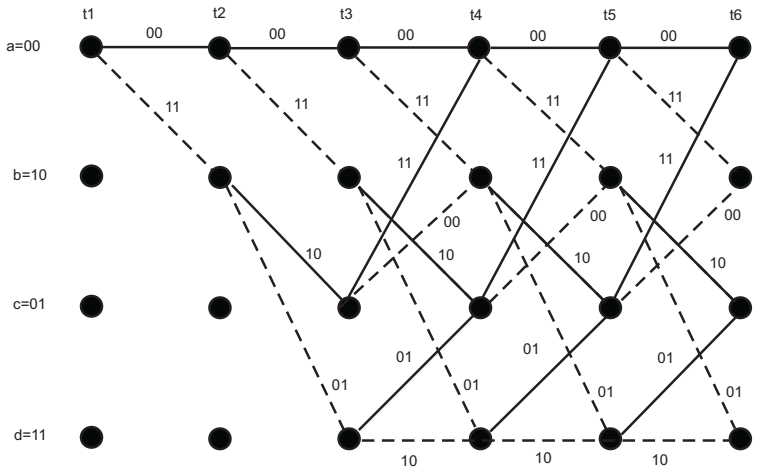

Figura 2.7: Diagrama de Trellis para un código convolucional.

transiciones que emergen de cada estado y tienen que ver con las dos posibilidades de entrada binaria existentes, '1' o '0'. La convención empleada es que un '1' lógico de entrada se representa con líneas de puntos, mientras que un '0' lógico de entrada se representa con líneas llenas.

Una característica importante de estos códigos se hace evidente de la observación del diagrama de estados. No es posible pasar de un estado a otro arbitrario. Esto resulta del mecanismo de memoria que el sistema tiene.

2.4.2. Trellis para un código convolucional

El diagrama de estados permite conocer la transición entre estados, pero no aporta información acerca de la historia del sistema, es decir, de los estados y transiciones iniciales [3, 8, 13].

El diagrama de Trellis [3] tiene en cuenta la dimensión del tiempo, poniendo en evidencia la forma en que la secuencia realmente sucede. El diagrama de Trellis para la secuencia de entrada $m = (100101)$ que entra al codificador convolucional se representa en la figura 2.7. Una propiedad importante de los códigos convolucionales es que, el sistema tiene, a partir de determinado momento, una repetición en su estructura. La estructura se repite después de K ramas, donde K es la constante de longitud asociada a estos códigos.

Las líneas sólidas representan un bit de entrada '0' mientras que líneas punteadas se usan para representar un bit de entrada '1'. Existen 2^{K-1} posibles estados para ser representados. También hay dos ramas que emergen de cada estado pertenecientes a las dos posibilidades de bits de entrada, y dos ramas que arriban a cada estado.

2.4.3. Decodificación por máxima similitud

Para una cierta secuencia de salida de un código convolucional $\mathbf{u}^{(m)}$ existe una secuencia recibida, denominada $\mathbf{z}$, que resulta de la aparición de errores por efecto del ruido sobre la secuencia original $\mathbf{u}^{(m)}$.

El decodificador óptimo compara las probabilidades condicionales $P(\mathbf{z}/\mathbf{u}^{(m)})$ de que la secuencia recibida $\mathbf{z}$ corresponda a una secuencia conocida $\mathbf{u}^{(m)}$ y toma la decisión de elegir la secuencia que tiene la mayor probabilidad de coincidir con $\mathbf{z}$:

$$P(\mathbf{z}/\mathbf{u}^{(m')}) = \underset{\mathbf{u}^{(m)}}{máx}\, P(\mathbf{z}/\mathbf{u}^{(m)}) \tag{2.48}$$

Este es el criterio de máxima similitud [3, 4]. Esta obviamente de acuerdo con el hecho intuitivo de asignar la secuencia conocida en el decodificador que más se parece a la recibida.

La aplicación de este criterio a los códigos convolucionales se enfrenta con el hecho de que un gran número de secuencias posibles debe ser considerada. Para una secuencia de L bits se tienen 2^L posibilidades. El decodificador de máxima similitud elige una secuencia $\mathbf{u}^{(m')}$ del conjunto posible, que tenga la mayor probabilidad de parecerse al vector recibido.

Si el canal no tiene memoria, el ruido es aditivo blanco y Gaussiano, por lo tanto, cada símbolo está afectado de una forma independiente de los otros. Para un código convolucional de velocidad $1/n$ la función probabilidad que mide la similitud respecto del vector recibido es:

$$\begin{aligned} P(\mathbf{z}/\mathbf{u}^{(m)}) &= \prod_{i=1}^{\infty} P(\mathbf{z}_i/\mathbf{u}_i^{(m)}) \\ &= \prod_{i=1}^{\infty} . \prod_{j=1}^{n} P(z_{ji}/u_{ji}^{(m)}) \end{aligned} \tag{2.49}$$

Donde $\mathbf{z}_i$ es la i-ésima rama de la secuencia recibida $\mathbf{z}$, $\mathbf{u}_i^{(m)}$ es la i-ésima rama de la secuencia de una palabra de código $\mathbf{u}^{(m)}$, z_{ji} es el j-ésimo símbolo de código de $\mathbf{z}_i$, y $u_{ji}^{(m)}$ es el j-ésimo símbolo de código de $\mathbf{u}_i^{(m)}$, donde cada rama está constituida por n símbolos de código. El proceso de decodificación consiste en la elección de una secuencia que maximice la función similitud. Este problema se analiza a continuación

2.4.4. Algoritmo de decodificación convolucional de Viterbi

El algoritmo de decodificación de Viterbi [13, 14, 15] realiza la detección por máxima similitud. Para eso utiliza las propiedades del Trellis de un código convolucional.

El algoritmo intenta reducir la complejidad del calculo evitando tener en cuenta la totalidad de las secuencias posibles. El procedimiento consiste en calcular la distancia

entre la señal recibida en el instante t_i y los caminos o ramas entrantes del Trellis en ese instante en el estado analizado. En la medida que este criterio se va aplicando se evalúa la secuencia que tiene la menor distancia respecto de la recibida, de forma que la secuencia de máxima similitud finalmente aparece.

El camino o secuencia de máxima similitud, que es al mismo tiempo, aquel que presenta la menor distancia respecto del recibido, se denomina camino o secuencia sobreviviente, la cual es almacenada. De esta forma, el algoritmo de Viterbi trata de encontrar el camino con mínimo error. Decir entonces que la secuencia es la de máxima similitud, es también decir que se trata de aquella secuencia que tiene la menor distancia respecto de la recibida.

2.5. Códigos de paridad de baja densidad

Los límites de la técnica de corrección de errores fueron establecidos por Shannon en su trabajo sobre el model matemático para las comunicaciones [6], y luego de este planteo surgieron una serie de técnicas de codificación que no encontraron un acercamiento notorio a los límites predichos hasta la aparición en 1993 de los códigos turbo [16]. Posteriormente David J. C. Mackay y R. M. Neal en 1996 [17, 18] redescubrieron los llamados códigos Gallager [19], que por años permanecieron ocultos [20], a pesar de su excelente funcionamiento.

Los códigos Gallager, también conocidos como códigos de paridad de baja densidad, que en inglés son descriptos como *Low-Density Parity-Check Codes* (LDPC), son códigos de bloques lineales, que se construyen sobre la base de una matriz de paridad **H** que tiene como característica ser de baja densidad (en inglés *Sparse Matrix*), es decir, en el caso binario contiene pocos elementos '1' dispersos entre un gran número de elementos '0'. El diseño propuesto por Gallager [19] presenta el método constructivo, el algoritmo probabilístico iterativo, y los elementos principales de su desarrollo; pero la carencia de procesamiento de cálculo poderoso para esa época (año 1962) no permitió mostrar la eficacia de estos códigos.

Los códigos LDPC fueron inicialmente concebidos mediante la construcción de una matriz con un número fijo de '1' en filas y columnas. En este caso se habla de códigos LDPC regulares. Sin embargo el número de '1' en filas y columnas puede ser variado en su construcción, en este caso se denominan códigos LDPC irregulares.

El funcionamiento de los códigos LDPC es comparable al de los códigos turbo. Versiones modificadas del esquema original, de estructura irregular, basadas en construcciones sobre campos de Galois extendidos $GF(8)$ [21], presentan un funcionamiento cercano al límite de Shannon, tal como lo hacen los códigos turbo. Una de las características comunes de los códigos LDPC y los códigos turbo es la existencia de un proceso pseudo-aleatorio que forma parte de la construcción de los mismos.

2.5.1. Diferentes formas sistemáticas para un código de bloques

Un código lineal sistemático de bloques $C_b(n,k)$ está especificado unívocamente por su matriz generadora **G** [9], y cuando el código tiene forma sistemática dicha matriz es de la forma:

$$\mathbf{G} = \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{k-1} \end{bmatrix} = \begin{bmatrix} p_{0,0} & p_{0,1} & \cdots & p_{0,n-k-1} & 1 & 0 & 0 & \cdots & 0 \\ p_{1,0} & p_{1,1} & \cdots & p_{1,n-k-1} & 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ p_{k-1,0} & p_{k-1,1} & \cdots & p_{k,n-k-1} & 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

La matriz $\mathbf{G}$ se puede escribir en forma abreviada como:

$$\mathbf{G} = \left[\begin{array}{cc} \mathbf{P} & \mathbf{I_k} \end{array} \right]$$

donde $\mathbf{P}$ es la sub-matriz de paridad e $\mathbf{I_k}$ la sub-matriz identidad de dimensión $k \times k$. En este esquema la palabra de mensaje aparece al final de la palabra codificada. La forma sistemática de la matriz de paridad $\mathbf{H}$ del código C_b generado por la matriz $\mathbf{G}$ es:

$$\mathbf{H} = \left[\begin{array}{ccccccccc} 1 & 0 & 0 & \cdots & 0 & p_{0,0} & p_{1,0} & \cdots & p_{k-1,0} \\ 0 & 1 & 0 & \cdots & 0 & p_{0,1} & p_{1,1} & \cdots & p_{k-1,1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & p_{0,n-k-1} & p_{1,n-k-1} & \cdots & p_{k-1,n-k-1} \end{array} \right] = \left[\begin{array}{cc} \mathbf{I_{n-k}} & \mathbf{P^T} \end{array} \right]$$

donde $\mathbf{P^T}$ es la transpuesta de la sub-matriz de paridad $\mathbf{P}$. La matriz $\mathbf{H}$ esta construida de manera que el producto interno entre un vector fila $\mathbf{g_i}$ de $\mathbf{G}$ y un vector fila $\mathbf{h_j}$ de $\mathbf{H}$ sean ortogonales. Esto tiene como consecuencia que:

$$\mathbf{G} \circ \mathbf{H^T} = \mathbf{0} \tag{2.50}$$

entonces:

$$\mathbf{G} \circ \mathbf{H^T} = \mathbf{m} \circ \mathbf{G} \circ \mathbf{H^T} = \mathbf{0} \tag{2.51}$$

Los códigos LDPC son generados o diseñados partiendo de la matriz $\mathbf{H}$ y luego obteniendo la matriz generadora $\mathbf{G}$. Para obtener el síndrome $\mathbf{s}$ a partir de $\mathbf{H}$ y no de $\mathbf{H^T}$, la palabra codificada se genera como $\mathbf{c} = \mathbf{G^T} \circ \mathbf{m}$. Dado que $\mathbf{G^T}$ es una matriz de dimensión $n \times k$, y el vector de mensaje $\mathbf{m}$ es un vector de dimensión $k \times 1$, el resultado es un vector de código $\mathbf{c}$ de dimensión $n \times 1$.

La decodificación por síndrome se basa en el cálculo del vector síndrome $\mathbf{s}$ de acuerdo a la expresión:

$$\mathbf{s} = \mathbf{H} \circ \mathbf{c} \tag{2.52}$$

Toda palabra válida del código debe cumplir con la condición: $\mathbf{s} = \mathbf{0}$, es decir:

$$\mathbf{s} = \mathbf{H} \circ \mathbf{G^T} = \mathbf{0} \tag{2.53}$$

entonces queda:

$$\mathbf{H} \circ \mathbf{G^T} = \mathbf{0} \tag{2.54}$$

Esta última ecuación pone en evidencia que la matriz $\mathbf{H}$ contiene toda la información del código, y en especial esta expresión será de utilidad para el diseño del esquema de decodificación iterativa basado en el algoritmo de suma-producto, como se verá más adelante.

2.5.2. Descripción de un código LDPC

Los códigos de paridad de baja densidad son códigos de bloques lineales y binarios en la mayoría de los casos. En estos códigos, existe una matriz de generación **G** que convierte a los vectores de mensaje **m** en vectores de código **c**. Asociada a la matriz **G** se encuentra la matriz paridad **H**, que tiene como característica que cada vector de del código debe cumplir con la condición $\mathbf{H} \circ \mathbf{c} = \mathbf{0}$.

La construcción de estos códigos se basa en el diseño de la matriz **H** correspondiente, cuya característica más importante es que sus elementos son mayoritariamente ceros. De acuerdo a la definición de Gallager [19] un código LDPC se define como $C_{LDPC}(n, s, v)$ donde n es la longitud de las palabras del código y la matriz **H** posee s unos '1' por columna y v unos '1' por fila, siendo normalmente $s \geq 3$.

Si las filas de la matriz **H** son independientes, entonces la tasa del código es $\frac{v-s}{v}$. De no cumplirse esta condición la tasa queda $\frac{n-s'}{n}$ donde s' es la dimensión del espacio fila de la matriz **H**. Si el código es de construcción irregular, los valores de s y/o de v no son fijos, entonces se reemplazan por sus valores promedios para calcular la tasa. En estos códigos, la probabilidad de error decrece exponencialmente con la longitud de la palabra de código [19], y la distancia del mismo aumenta al aumentar su longitud.

2.5.3. Construcción de un código LDPC

Códigos LDPC regulares

El método constructivo fué planteado por Gallager [19] y consiste en fijar el número de unos sobre filas y columnas de manera de realizar una construcción aleatoria de la misma, conservando estas condiciones Las condiciones utilizadas en la construcción de la matriz **H** de un código LDPC regular y binario son [22]:

- La matriz **H** debe poseer un número constante v de unos por fila
- La matriz **H** debe poseer un número constante s de unos por columna
- El solapamiento de unos por fila y por columna sea a lo sumo de un uno. Esta condición garantiza la ausencia de ciclos en el correspondiente grafo bipartito.
- Las constantes s y v sean pequeñas respecto del tamaño de la palabra de código.

No es posible sin embargo exigir que se cumpla la tercera condición si se pretende construir códigos LDPC con óptimos o buenos parámetros, dado que es inevitable que el grafo bipartito de un código LDPC de alto rendimiento contenga ciclos [22].
Normalmente, luego de esta contrucción la matriz **H** resultante no estará en formato sistemático, por lo tanto es conveniente mediante operaciones sobre las filas,

utilizando el método de eliminación de Gauss, dar forma a una matriz equivalente $\mathbf{H'} = [\mathbf{I_{n-k}}\ \mathbf{P^T}]$, donde $\mathbf{I_{n-k}}$ es la matriz identidad de dimensión$(n-k) \times (n-k)$. La matriz inicialmente diseñada será entonces la matriz $\mathbf{H}$ del código LDPC, para la cual existe una matriz $\mathbf{G}$ de la forma $\mathbf{G} = [\mathbf{P}\ \mathbf{I_k}]$.

El método consiste entonces en dar forma a una matriz $\mathbf{H} = [\mathbf{A}\ \mathbf{B}]$ que normalmente estará en forma no sistemática. Como $\mathbf{H}$ es de baja densidad, las submatrices $\mathbf{A}$ y $\mathbf{B}$ también lo son. La sub-matriz $\mathbf{A}$ es una matriz cuadrada de dimensión $(n-k) \times (n-k)$ que de no ser singular poseerá una matriz inversa $\mathbf{A^{-1}}$. La submatriz $\mathbf{B}$ es de dimensión $(n-k) \times k$.

Cuando se aplica el método de eliminación de Gauss sobre la matriz $\mathbf{H} = [\mathbf{A}\ \mathbf{B}]$, utilizando operaciones en el campo binario, se llega a la matriz $\mathbf{H'} = [\mathbf{I_k}\ \mathbf{A^{-1}}\ \mathbf{B}] = [\mathbf{I_k}\ \mathbf{P^T}]$. Esta operación equivale a premultiplicar a la matriz $\mathbf{H} = [\mathbf{A}\ \mathbf{B}]$ por $\mathbf{A^{-1}}$.

Una vez formada la matriz $\mathbf{H'}$ del código LDPC, se puede construir la matriz generadora $\mathbf{G}$ empleando las sub-matrices calculadas $\mathbf{G} = [\mathbf{P}\ \mathbf{I_k}]$. Sin embargo la matriz de utilidad es la $\mathbf{H}$.

En general y en función del método constructivo de la correspondiente matriz $\mathbf{H}$, los códigos LDPC pueden clasificarse en [23]:

- Códigos de construcción aleatoria
- Códigos LDPC de construcción estructurada

Aun cuando los códigos de construcción aleatoria pueden tener mejor funcionamiento que los de construcción estructurada, estos últimos son mas adecuados en términos de la complejidad constructiva del codificador. La contrucción propuesta por D. Mackay [18] por ejemplo es de naturaleza aleatoria, mientras otros enfoques como el de Shu Lin [24] se basan en consideraciones geométricas, o bien otros en la conformación de la matriz con propiedades cíclicas o cuasi-cíclicas. También existe otro planteo dentro de los códigos LDPC de construcción estructurada que se basa en matemática combinatoria (denominada diseño en bloques incompletos balanceados) [23].

Códigos LDPC irregulares

En estos códigos, el número de unos por fila y/o columna puede variar. En general se encuentra que los códigos irregulares pueden funcionar mejor que los regulares. También existen diferentes métodos [22] para la construcción de códigos irregulares.

2.5.4. Decodificación de los códigos LDPC

El vector codificado $\mathbf{c}$ se obtiene a partir del mensaje $\mathbf{m}$ por medio de la operación matricial $\mathbf{c} = \mathbf{G^T} \circ \mathbf{m}$ donde $\mathbf{G^T} = \begin{bmatrix} \mathbf{P^T} \\ \mathbf{I_k} \end{bmatrix}$. Al transmitir el vector $\mathbf{c}$, éste es afectado

por el ruido del canal, entonces el vector recibido resulta ser $\mathbf{r} = \mathbf{c} + \mathbf{n}$ donde $\mathbf{n}$ es ruido blanco Gaussiano. En el receptor, el síndrome $\mathbf{s}$ se calcula como $\mathbf{s} = \mathbf{H} \circ \mathbf{r} = \mathbf{H} \circ (\mathbf{G^T} \circ \mathbf{m}) + \mathbf{n} = \mathbf{H} \circ \mathbf{m}$.

Luego, la esencia de la decodificación, es encontrar un vector $\hat{\mathbf{d}}$, que es una estimación del vector $\mathbf{c}$, que cumpla con la condición $\mathbf{H} \circ \hat{\mathbf{d}} = \mathbf{0}$.

Para realizar esta estimación se utiliza el algoritmo de suma-producto o algoritmo de propagación de creencias [17]. Este algoritmo estima la probabilidad a posteriori de cada símbolo en función de la señal recibida, de las ecuaciones de paridad y de las características del canal. El algoritmo suma-producto se describe sobre la base de un grafo bipartito que se define de acuerdo a la matriz $\mathbf{H}$ en el cual se grafica la relación entre dos tipos de nodos:

- Los nodos representativos de los símbolos de transmisión, los cuales se conocen como nodos símbolos.
- Los nodos representativos de las ecuaciones de paridad que vinculan a los símbolos, los cuales se conocen como nodos paridad.

Las filas de la matriz $\mathbf{H}$ describen a los símbolos que se encuentran involucrados en determinada ecuación de paridad. De esta forma, la conexión entre el nodo símbolo d_j y la ecuación de paridad h_i existirá siempre y cuando el elemento de matriz $\mathbf{H}$ sea uno, es decir, $H_{ij} = 1$.

El estado de cada nodo paridad depende del los estados de los nodos símbolos a los cuales está vinculado. Los nodos de paridad vinculados a un nodo símbolo se denominan nodos hijos y los nodos símbolos vinculados a uno de paridad se denominan nodos padres. La figura 2.8 ilustra el grafo bipartito [25] definido para una matriz $\mathbf{H}$ de $m \times n$ y un vector símbolo $\mathbf{d}$ de $n \times 1$. En el algoritmo de suma-producto cada

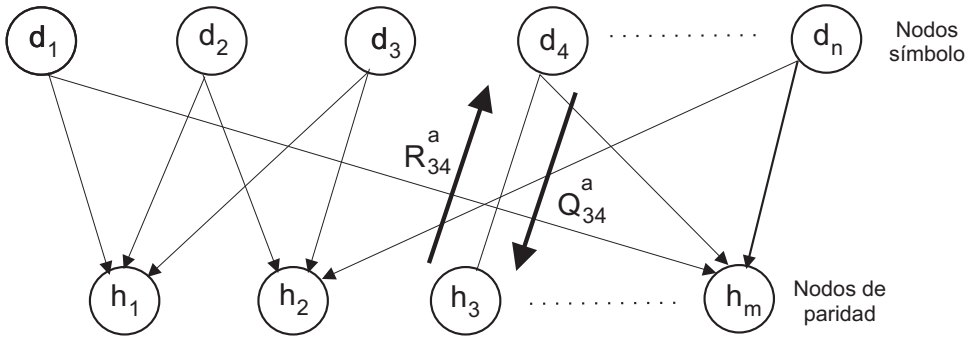

Figura 2.8: Grafo bipartito. Nodos símbolo y de paridad.

nodo símbolo d_j envía al nodo hijo de paridad h_i la información probabilística Q_{ij}^a. El valor de Q_{ij}^a indica la probabilidad de que el nodo de paridad i se encuentre en el

estado a, basado en la información proporcionada por los otros nodos de paridad que están vinculados con el nodo símbolo j.

Por otra parte, cada nodo de paridad h_i envía la información R_{ij}^a a cada nodo símbolo padre d_j. El valor R_{ij}^a informa la probabilidad de que la paridad del nodo i se satisfaga, suponiendo que le nodo símbolo j se encuentra en el estado a, de acuerdo a la información proporcionada por los otros nodos símbolos relacionados con el nodo de paridad i. Este proceso de intercambio de información entre nodos es iterativo.

Este proceso se detiene si la condición de síndrome es satisfecha, con lo cual se procede a una decodificación, o bien, si el número de iteraciones establecido es alcanzado sin que la condición de síndrome se cumpla, generando en este caso una palabra decodificada donde cada símbolo de la misma ha sido decodificado de manera optima.

2.5.5. Descripción del algoritmo de suma-producto

En este algoritmo los valores de Q_{ij}^a se ajustan en un principio a la probabilidad a priori de los símbolos f_j^a, que es la probabilidad de que el j ésimo símbolo sea a, con $a = \{0, 1\}$. Esta información es dependiente del canal que se emplee en el modelo. Para el caso de un canal Gaussiano, esta probabilidad se estima con la función distribución de probabilidad Gaussiana.

Luego de este proceso inicial, se evalúan los mensajes entre nodos en forma iterativa. El mensaje R_{ij}^a que cada nodo de paridad i envía a sus nodos padres j es la probabilidad de que el nodo de paridad i se satisface cuando el nodo padre está en el estado a. Que el nodo se satisfaga significa que está de acuerdo con la ecuación del síndrome del nodo h_i. Éste valor es igual a:

$$p(h_i/d_j = a) = \sum_{\mathbf{d}:d_j=a} p(h_i/\mathbf{d}) \cdot p(\mathbf{d}/d_j = a) \tag{2.55}$$

La probabilidad es calculada sobre todos los valores del vector decodificado $\mathbf{d}$ para los que el control de paridad se satisface con la condición de que el nodo padre esté en el valor a. Para el nodo de paridad h_i la información actualizada a transmitir al nodo símbolo d_j se calcula para cada valor de estado a y es igual a:

$$R_{ij}^a = \sum_{\mathbf{d}:dj=a} p(h_i/\mathbf{d}) \cdot \prod_{k \in N(i)\backslash j} Q_{ik}^{d_k} \tag{2.56}$$

En esta expresión, $N(i)$ representa el conjunto de subíndices de todos los nodos padre del nodo de paridad h_i, mientras que la notación $N(i)\backslash j$ indica la exclusión del nodo j de ese conjunto. La probabilidad $p(h_i/\mathbf{d})$ de que el control de paridad se satisfaga es 0 ó 1 para un dado valor de $\mathbf{d}$.

El mensaje probabilístico Q_{ij}^a que el nodo símbolo j envía a sus nodos hijo de paridad i, es la estimación que el nodo tiene de que está en el estado a de acuerdo a la información proveniente de los otros nodos hijo. Luego, y utilizando la regla de Bayes:

$$p(d_j = a/\{h_i\}_{i\in M(j)\backslash i}) = \frac{p(d_j = a)p(\{h_i\}_{i\in M(j)\backslash i}/d_j = a)}{p(\{h_i\}_{i\in M(j)\backslash i})} \tag{2.57}$$

La información que el nodo símbolo j envía a sus nodos hijo de paridad i es entonces:

$$Q_{ij}^a = \alpha_{ij} \cdot f_j^a \cdot \prod_{k\in M(j)\backslash i} R_{kj}^a \tag{2.58}$$

donde $M(j)$ representa el conjunto de subíndices de todos los nodos hijo del nodo símbolo d_j, mientras que la notación $M(j)\backslash i$ indica la exclusión del nodo i de ese conjunto.

El factor f_j^a es la probabilidad a priori de que d_j esté en el estado a. La constante de normalización α_{ij} se evalúa de forma que se cumpla $\sum_a Q_{ij}^a = 1$.

De esta manera, el conocimiento del valor de los coeficientes Q_{ij}^a permite determinar los valores de los coeficientes R_{ij}^a con los cuales se puede realizar una estimación del valor del símbolo en la posición j:

$$\hat{d}_j = arg\,max\, f_j^a \cdot \prod_{k\in M(j)} R_{kj}^a \tag{2.59}$$

Si este vector decodificado y estimado confirma la ecuación de síndrome $\mathbf{S} = \mathbf{H} \circ \hat{\mathbf{d}} = \mathbf{0}$ entonces se considera vector válido de código y la decodificación acertada. En caso de no suceder esto último, la decodificación optimizará de acuerdo al criterio máxima probabilidad a posteriori MPA [9] la estimación hecha sobre cada bit, pero no necesariamente se producirá la decodificación acertada de la palabra de código que había sido efectivamente transmitida.

2.5.6. Algoritmo de Mackay-Neal

Los códigos LDPC fueron descubiertos por Gallager en 1962 [19] y fueron recientemente redescubiertos por [18]. Este algoritmo de decodificación estima la probabilidad de cada símbolo en función de la señal recibida y de las características del canal. El algoritmo se describe sobre la base de un grafo bipartito que se define de acuerdo a la matriz **H** en el cual se gráfica la relación entre dos tipos de nodos, los nodos d_j representativos de los símbolos de recepción y los nodos h_i representativos de las ecuaciones de paridad que vinculan a los símbolos (ver figura 2.8).

Las filas de la matriz **H** describen a los símbolos que se encuentran involucrados con una determinada ecuación de paridad. De esta forma, la conexión entre el nodo símbolo d_j y el nodo paridad h_i existirá siempre y cuando el elemento de la matriz **H** sea uno, es decir $H_{ij} = 1$. El estado de un nodo de paridad depende de los nodos símbolos a los cuales está vinculado.

En este algoritmo, cada nodo símbolo d_j envía al nodo paridad h_i la información probabilística Q_{ij}^a basada en la información proporcionada por los otros nodos paridad relacionados con ese nodo símbolo, de que el nodo paridad se encuentra en el estado a. Por otra parte, cada nodo paridad h_i envía la información R_{ij}^a informando la probabilidad de que la paridad del nodo i se satisfaga, suponiendo que el nodo símbolo j se encuentra en el estado a, de acuerdo a la información proporcionada por los restantes nodos símbolos que están relacionados con h_i.

A continuación se detalla el algoritmo, el cual consta de las siguientes etapas:

Inicialización

Los valores de Q_{ij}^a se ajustan en un principio a la probabilidad a priori f_j^a de los símbolos, que es la probabilidad de que el j ésimo símbolo sea a. En el caso binario, $a = \{0, 1\}$, por tanto, las variables Q_{ij}^0 y Q_{ij}^1 son inicializadas con los valores f_j^0 y f_j^1 respectivamente. Éstos se calculan como [18]:

$$f_j^1 = \frac{1}{1 + e^{-\frac{2y_j}{\sigma^2}}} \tag{2.60}$$

y

$$f_j^0 = 1 - f_j^1 = \frac{e^{-\frac{2y_j}{\sigma^2}}}{1 + e^{-\frac{2y_j}{\sigma^2}}} \tag{2.61}$$

donde y_j es el valor del j ésimo símbolo a la salida del canal afectado con ruido blanco Gaussiano N_0 de varianza $\sigma^2 = N_0/2$.

Paso horizontal

Se define $\delta Q_{ij} = Q_{ij}^0 - Q_{ij}^1$ y se calcula para cada i, j:

$$\delta R_{ij} = \prod_{j' \in N(i) \backslash j} \delta Q_{ij'} \tag{2.62}$$

donde $N(i)$ representa el conjunto de subíndices de todos los nodos símbolos d_j que participan del nodo paridad h_i, mientras que $N(i)\backslash j$ indica la exclusión del nodo j de ese conjunto. Con el valor δR_{ij} hallado se obtiene entonces el conjunto:

$$R_{ij}^0 = {}^1/_2 \cdot (1 + \delta R_{ij}) \tag{2.63}$$

$$R_{ij}^1 = {}^1/_2 \cdot (1 - \delta R_{ij}) \tag{2.64}$$

Paso vertical

Para cada i, j con $a = \{0, 1\}$ se estima:

$$Q_{ij}^a = \alpha_{ij} \cdot f_j^a \cdot \prod_{i' \in M(j) \backslash i} R_{i'j}^a \tag{2.65}$$

donde $M(j)$ representa el conjunto de subíndices de todos los nodos paridad h_i que participan del nodo símbolo d_j. La constante α_{ij} se elije de forma tal que cumpla con la condición $Q_{ij}^0 + Q_{ij}^1 = 1$. Para facilitar esta última condición se define la variable C_{ij}^a como:

$$C_{ij}^a = f_j^a \cdot \prod_{i' \in M(j) \backslash i} R_{i'j}^a \tag{2.66}$$

la cual toma los valores:

$$C_{ij}^0 = f_j^0 \cdot \prod_{i' \in M(j) \backslash i} R_{i'j}^0 \tag{2.67}$$

$$C_{ij}^1 = f_j^1 \cdot \prod_{i' \in M(j) \backslash i} R_{i'j}^1 \tag{2.68}$$

entonces Q_{ij}^a queda:

$$Q_{ij}^a = \alpha_{ij} \cdot C_{ij}^a \tag{2.69}$$

como se debe cumplir la condición $Q_{ij}^0 + Q_{ij}^1 = 1$, de la ecuación 2.69 se tiene:

$$1 = \alpha_{ij} \cdot C_{ij}^0 + \alpha_{ij} \cdot C_{ij}^1 \tag{2.70}$$

por tanto α_{ij} resulta ser:

$$\alpha_{ij} = \frac{1}{C_{ij}^0 + C_{ij}^1} \tag{2.71}$$

y obteniendo de las ecuaciones 2.69 y 2.71:

$$Q_{ij}^0 = \frac{C_{ij}^0}{C_{ij}^0 + C_{ij}^1} \tag{2.72}$$

y

$$Q_{ij}^1 = \frac{C_{ij}^1}{C_{ij}^0 + C_{ij}^1} \tag{2.73}$$

Estimación a posteriori

Luego para cada j se realiza una estimación a posteriori de Q_j^0 y Q_j^1 usando:

$$Q_j^a = \alpha_j \cdot f_j^a \cdot \prod_{i \in M(j)} R_{ij}^a \tag{2.74}$$

donde en forma similar α_j se elige para que cumpla con la condición $Q_j^0 + Q_j^1 = 1$. Si al término $\prod R_{ij}^a$ de la ecuación 2.74, se descompone como:

$$\prod_{i \in M(j)} R_{ij}^a = (\prod_{i' \in M(j) \setminus i} R_{i'j}^a) \cdot R_{ij}^a \tag{2.75}$$

entonces, la ecuación 2.74 queda:

$$Q_j^a = \alpha_j \cdot f_j^a \cdot (\prod_{i' \in M(j) \setminus i} R_{i'j}^a) \cdot R_{ij}^a \tag{2.76}$$

y usando C_{ij}^a definida en la ecuación 2.66 se tiene:

$$Q_j^a = \alpha_j \cdot C_{ij}^a \cdot R_{ij}^a \tag{2.77}$$

De nuevo, como se debe cumplir con la condición $Q_j^0 + Q_j^1 = 1$, de la ecuación 2.77 surge:

$$1 = \alpha_j \cdot C_{ij}^0 \cdot R_{ij}^0 + \alpha_j \cdot C_{ij}^1 \cdot R_{ij}^1 \tag{2.78}$$

resultando entonces α_j:

$$\alpha_j = \frac{1}{C_{ij}^0 \cdot R_{ij}^0 + C_{ij}^1 \cdot R_{ij}^1} \tag{2.79}$$

entonces de las ecuaciones 2.77 y 2.79 se obtiene:

$$Q_j^0 = \frac{C_{ij}^0 \cdot R_{ij}^0}{C_{ij}^0 \cdot R_{ij}^0 + C_{ij}^1 \cdot R_{ij}^1} \tag{2.80}$$

y

$$Q_j^1 = \frac{C_{ij}^1 \cdot R_{ij}^1}{C_{ij}^0 \cdot R_{ij}^0 + C_{ij}^1 \cdot R_{ij}^1} \tag{2.81}$$

Estos valores son utilizados para obtener un valor tentativo para cada símbolo d_j, usando $\hat{d}_j = max(Q_j^a)$. En forma práctica se tiene:

$$\hat{d}_j = 0 \quad si \quad Q_j^0 > Q_j^1 \quad sino \quad \hat{d}_j = 1 \tag{2.82}$$

Si se cumple que $\mathbf{H} \circ \hat{\mathbf{d}} = \mathbf{0}$ se detiene el algoritmo, sino, se repite a partir del paso horizontal.

2.5.7. Análisis bibliográfico de los decodificadores LDPC

Como se indicó en el capítulo 1, la presente tesis está principalmente orientada a la implementación en dispositivos lógicos programables de códigos correctores de errores. En las implementaciones de éstos códigos normalmente el decodificador presenta una complejidad mucho mayor que el correspondiente codificador, como es el caso de los códigos LDPC. Particularmente en esta sección se da una idea sobre el estado del arte de la implementación de decodificadores LDPC.

1. **Decodificador suma-producto o Mackay-Neal**: El algoritmo suma-producto es la forma más básica de implementar la decodificación iterativa, característica de los códigos LDPC. Tal como su nombre lo indica este algoritmo utiliza una gran cantidad de productos y cocientes, y por otra parte se asume que se trabaja con aritmética de punto flotante y con precisión infinita. En términos generales, la bibliografía consultada desaconseja utilizar este algoritmo cuando se desea implementar en forma circuital un decodificador LDPC, proponiendo entonces diferentes simplificaciones sobre el mismo. Particularmente en [26, 27, 28] se tratan los efectos que tiene el recorte y la cuantización de los niveles de entrada en la performance del algoritmo de decodificación. Se advierte que en los decodificadores, que emplean principalmente multiplicadores y cocientes, el error de cuantización puede producir divisiones por cero, o que el resultado de un producto sea siempre cero. Se demuestra también que al eliminar en el algoritmo de decodificación los productos y cocientes, el decodificador se vuelve más robusto con respecto al error de cuantización.

2. **Decodificador paralelo**: Otra forma posible de implementar decodificadores LDPC se describe en [29, 30] mediante el uso de un decodificador paralelo, en el cual cada nodo símbolo o de control de paridad, está implementado de forma individual como una unidad. Todas estas unidades están conectadas a través de una red de interconexión, reflejando las conexiones del grafo bipartito, similar al que aparece en la figura 2.8. Con este tipo de decodificadores se puede lograr gran velocidad. Sin embargo su implementación es muy compleja, especialmente cuando se tienen muchos nodos símbolos y de control de paridad. Este tipo de enfoque está limitado a decodificadores que usen una matriz de control de paridad **H** de tamaño reducido. Tampoco permite variar en forma sencilla el tipo de matriz de chequeo de paridad **H**.

3. **Decodificador basado en la partición de la matriz de chequeo de paridad H**: Éste es un enfoque que aparece en [31, 32, 33], que permite trabajar con matrices de control de paridad de mayor tamaño, sin incrementar mucho el nivel de complejidad en la implementación. La idea principal en estos mecanismos de

simplificación es la de utilizar una matriz de control de paridad que permite ser subdividida en varias submatrices que se repiten periódicamente. La limitación de este tipo de enfoque es que no permite trabajar con cualquier tipo de matriz de control de paridad, ya que la misma debe cumplir la condición de repetición periódica.

4. **Decodificadores que no usan productos ni cocientes**: En [34] se tiene una primera aproximación de como pasar de un decodificador que emplea multiplicadores y cocientes a uno que emplea sumas y restas. Si U y V son dos variables aleatorias binarias, y $L(U) = P(U = 0)/P(U = 1)$, obtiene:

$$\begin{aligned} L(U \oplus V) &= \text{máx}[0, (L(U) + L(V)] - \text{máx}[L(U), L(V)] \\ &+ \ln(1 + e^{-|L(U)+L(V)|}) \\ &- \ln(1 + e^{-|L(U)-L(V)|}) \end{aligned}$$

Como se puede observar es necesario emplear un factor de corrección en los cálculos. Este factor puede ser implementado mediante dos tablas de búsqueda o utilizando una aproximación lineal por tramos.

En [35] se realiza un análisis previo en cual se ilustra porque la implementación de un decodificador de suma producto es muy difícil de implementar directamente. Basándose en este análisis, se proponen dos decodificadores, uno serie y otro paralelo en los cuales se disminuye la complejidad de la implementación. El decodificador serie ya se presentó en el artículo tratado anteriormente. El factor determinante, es la implementación de la función que produce el factor de corrección. Muestra que se puede utilizar un valor fijo, el cual se obtiene por prueba y error mediante simulación. Para hallar este valor define a $g(x_i) = \ln(1 + e^{|x_i|})$, donde $x_1 = L(U) + L(V)$ y $x_2 = L(U) - L(V)$. De acuerdo a estas definiciones, el factor de corrección queda dado por $g(x_1) - g(x_2)$ y para hallarlo en forma de un valor constante c, propone:

$$\begin{aligned} g(x_1) - g(x_2) &= +c \; si \; |x_1| < 2 \; y \; |x_2| > 2|x_1| \\ &= -c \; si \; |x_2| < 2 \; y \; |x_1| > 2|x_2| \\ &= 0 \; el \; resto \end{aligned}$$

El valor obtenido sólo es válido para la matriz de chequeo de paridad $\mathbf{H}$ usada en el cálculo. Al igual que en [34], otra solución propuesta es el uso de una aproximación lineal por tramos, y por ultimo el uso de tablas de búsqueda. En el decodificador paralelo si bien utiliza un solo factor de corrección $h(x) = \ln|e^x - 1|$, tiene el inconveniente que $h(x)$ se aproxima a $-\infty$ cuando x se acerca a 0. Este comportamiento hace difícil su implementación en tablas de búsqueda.

En [36] se analizan varios tipos de aproximaciones para decodificar códigos LDPC. Se hace notar que al implementar un decodificador LDPC, éste ya no trata con aritmética de punto flotante, ni con precisión infinita. Una técnica sugerida para facilitar las implementaciones, es la de eliminar los productos y cocientes; lo cual se logra aplicando logaritmos a la relaciones de probabilidad (*likelihood ratios*). La primer aproximación propuesta, utiliza la relación tangente hiperbólica *tanh*, que si bien elimina el uso de multiplicadores y cocientes, requiere poder evaluar esta relación hiperbólica, lo cual es difícil de implementar.

Otro procedimiento propuesto elimina el uso de la tangente hiperbólica y la reemplaza por la operación $f(f(a)+f(b))$ donde la función $f(x) = ln(\frac{e^x+1}{e^x-1})$ es una transformada, que tiene la propiedad que $f(f(x)) = x$. Esta transformada puede ser fácilmente implementada en *software*, pero su implementación en circuitos lógicos digitales no resulta una tarea sencilla.

Posteriormente, y mediante el uso de logaritmos Jacobianos, se reemplaza a la tangente hiperbólica por dos funciones logarítmicas factibles de ser implementadas en tablas de búsqueda. El algoritmo propuesto es similar al definido en [34]:

$$\begin{aligned} L(U \oplus V) &= sign(L(U))sign(L(V))\,\text{mín}[L(U), L(V)] \\ &+ \ln(1 + e^{-|L(U)+L(V)|}) \\ &- \ln(1 + e^{-|L(U)-L(V)|}) \end{aligned}$$

Utilizando el procedimiento para hallar un factor de corrección fijo c, propuesto en [35], se simplifica el algoritmo obteniendo:

$$L(U \oplus V) \sim sign(L(U))sign(L(V))\,\text{mín}[L(U), L(V)] - c$$

En este caso para cada código hay que hallar el mejor factor de corrección.

Por último en [37] se proponen dos versiones simplificadas del algoritmo de propagación de creencias. Los algoritmos presentados no utilizan factor de corrección, en vez de ello se recurre a realizar ciertas partes del algoritmo usando decisión rígida y en otras partes utiliza decisión no rígida.

Como ejemplo de la mezcla de operaciones de decisión rígida y no rígida, se muestra el primer algoritmo propuesto, ya que los dos son similares. Se define la variable σ_m como el resultado que se tiene en el nodo paridad m cuando se evalúa en forma rígida la probabilidad a posteriori q_n, y se define a $\overline{\sigma}_m$ como su complemento en módulo-2. La secuencia de pasos del algoritmo es la siguiente:

- Inicialización: se utilizan las variables $\hat{x}_n$ y $|r_n|$. $\hat{x}_n$ se inicializa con el valor rígido del símbolo recibido y_n y $|r_n|$ con $|y_n|$
- Paso 1: para cada n y m se evalúan:

$$\sigma_n = \sum_{n' \in N(m)} \hat{x'}_n [mod 2]$$

$$|y_{nm}|_{min} = \min_{n' \in N(m) \backslash n} |y'_n|$$

- Paso 2: para cada n calcula

$$z_n = |r_n| + \sum_{m \in M(n)} (\overline{\sigma}_m - \sigma_m)|y_{nm}|_{min}$$

- Paso 3: si $z_n < 0$ entonces $\hat{x}_n = \hat{x}_n \oplus 1$, luego se repite el Paso 1

Si bien este algoritmo no utiliza tablas de búsqueda, reduciendo por consiguiente, su complejidad circuital, se necesita realizar entre 50 y 200 iteraciones del algoritmo para obtener la misma performance que se obtiene con solamente 16 iteraciones, cuando se usan tablas de búsqueda. Como se ve se tiene una relación de compromiso entre la complejidad de *hardware* y la velocidad del decodificador

Al emprender la implementación circuital de un decodificador es deseable que este puede trabajar con cualquier tamaño y tipo de matriz de paridad **H** (incluidas las generadas aleatoriamente, las generadas con una determinada regla de construcción, como las quasi-cíclicas, sean de tipo sistemático, o no sistemático) y que su arquitectura puede ser fácilmente configurada para diferentes tasas de código.

De la bibliografía consultada, surge que para poder lograr tales requerimientos es necesario utilizar el último criterio, es decir usar algoritmos que eviten el uso de multiplicadores y cocientes. Al evaluar los diversos algoritmos presentados, surgen las siguientes consideraciones:

- En la simplificación de los algoritmos no siempre un incremento en la sencillez matemática del mismo, se corresponde con un incremento en la sencillez de su implementación.
- En algunos casos, las simplificaciones algorítmicas de los códigos presentan una performance bastante degradada respecto de la que se obtiene usando el algoritmo de suma producto en su modelo teórico.
- La mayoría de los trabajos no describen totalmente el algoritmo completo.

- Entre las simplificaciones más aceptables, es necesario el uso de un factor de corrección, el cual puede ser implementado de diferentes formas:
  - Utilizando un valor de corrección fijo, lo cual implica que solo sirve para una determinada matriz de control de paridad H.
  - Haciendo uso de una aproximación lineal por tramos, que no siempre es fácil de implementar físicamente.
  - Utilizando una tabla de búsqueda, que puede ser implementada fácilmente en una memoria.

En base a estas consideraciones, se optó por el desarrollo de un algoritmo de decodificación que sea sencillo de implementar, tenga una buena performance con respecto al algoritmo de suma-producto teórico, no dependa de la tecnología en la cual vaya a ser implementado, y permita el uso de cualquier matriz de control de paridad **H**.

Una ventaja adicional surge en [25] donde se presenta una descripción gráfica de la factorización de una función global, en el producto de funciones locales. Mediante cálculos sencillos permite que el algoritmo de suma-producto calcule varias funciones marginales derivadas de una función global. Una amplia variedad de algoritmos usados en inteligencia artificial, procesamiento de señal y comunicaciones digitales pueden ser obtenidos a partir del algoritmo de suma-producto. Esto incluye al algoritmo de Viterbi y al código turbo iterativo.

Como puede verse de este último artículo resulta muy interesante el poder implementar un algoritmo en forma física, que cumpla las mismas funciones que el algoritmo de suma-producto ideal y con una performance en su funcionamiento similar.

2.6. Códigos turbo

El esquema de control de errores original presentado por Berrou, Glavieux y Thitimajshima [38] en 1993 revolucionó el tema de la decodificación proponiendo un mecanismo iterativo que proporciona ganancias de código óptimas, las cuales llevan al sistema a trabajar en cercanía del límite de Shannon.

En esta estructura inicial dos codificadores convolucionales se ordenan en una configuración de concatenado paralelo involucrando también un permutador de datos pseudo aleatorio [39, 40, 41], de manera que cada dato de entrada es codificado dos veces con la ayuda del permutador aleatorio, cuyo efecto es el de independizar estadísticamente las dos secuencias de datos generadas. En este sentido se prefiere denominar de ahora en adelante permutador al entrelazador de datos que es utilizado clásicamente en un código turbo.

En la forma más usual, la estructura del codificador turbo esta basada en la utilización de dos codificadores convolucionales. Estos son construidos con máquinas secuenciales de estados finitos de respuesta impulsiva infinita, conocidos como codificadores convolucionales sistemáticos recursivos (CCSR), que generalmente son de tasa $k/n = 1/2$. Frecuentemente se emplea la técnica de eliminación selectiva de salidas, donde alguno de los datos de redundancia es omitido de acuerdo a una ley conocida que permite entonces mejorar la tasa del código.

En el proceso iterativo de decodificación la información comunicada de uno a otro decodificador es una estimación de los datos, de manera que los decodificadores empleados en estos esquemas deben operar con estimaciones a la entrada y a la salida del sistema, aplicando lo que se conoce como decisión suave o no rígida sobre la información.

En el mecanismo iterativo, el primer decodificador adopta un conjunto de estimaciones para los bits a decodificar y comunica lo que se llama una salida extrínseca o información extrínseca que será proporcionada al segundo decodificador, que emplea la información como una información a priori, que conjuntamente con la información propuesta por los bits de entrada y la estructura de codificación, le permite realizar su estimación y proporcionar su información extrínseca, que será empleada por el primer decodificador como información a priori para su nueva estimación.

Esta información proveniente del segundo decodificador es empleada por el primer decodificador para generar una estimación más adecuada, permitiéndole así corregir más errores de los que había corregido en primera instancia. En forma iterativa este procedimiento genera una estimación cada vez mas adecuada de la información transmitida, pero normalmente la mejora en la estimación decrece con el número de iteraciones, haciendo que dicho número se establezca en un valor razonable donde la mejora generada no se incrementa tanto para justificar la continuación del proceso

iterativo.

Como procedimiento de decodificación que procesa estimaciones de entrada y salida aparece el algoritmo de Máxima Probabilidad A Posteriori (MPA) conocido como algoritmo BCJR [42]. Posteriores optimizaciones de este algoritmo llevaron a la aplicación de algoritmos similares de menor complejidad, como el algoritmo de Viterbi con decisión suave, que en inglés se conoce como *Soft Output Viterbi Algorithm* (SOVA), y el algoritmo MPA con estimaciones logarítmicas, llamado normalmente algoritmo LOG MPA.

2.6.1. Codificador turbo

La estructura de un codificador turbo obedece a la de dos codificadores organizados en concatenación paralela y asistidos por un bloque de permutación de datos [9, 38, 40, 43], así como de un procesador de multiplexado y eliminación selectiva de salidas [39, 44], tal como el que se ve en la figura 2.9. En la estructura más tradicional los

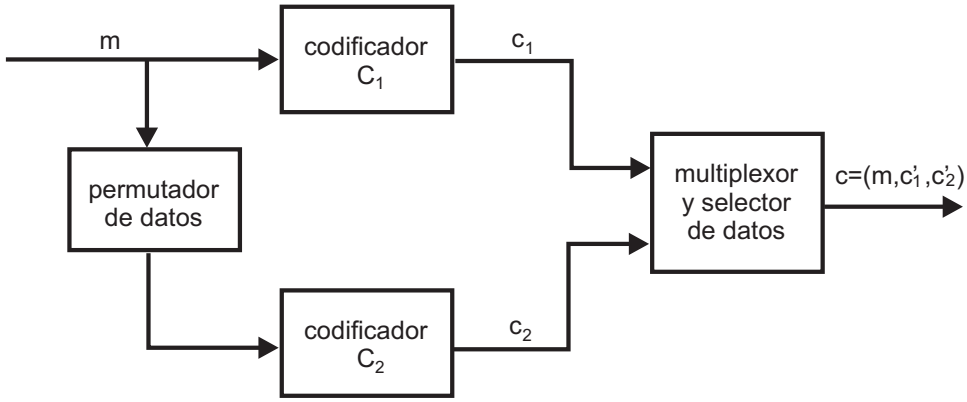

Figura 2.9: Esquema del codificador turbo.

codificadores en la figura denominados C_1 y C_2 son CCSR [38, 40] de tasa $k/n = 1/2$ de manera que $\mathbf{c_1'} = \mathbf{c_1}$, $\mathbf{c_2'} = \mathbf{c_2}$ y la longitud de las secuencias $\mathbf{c_1'}$ y $\mathbf{c_2'}$ es la mitad de las secuencias $\mathbf{c_1}$ y $\mathbf{c_2}$ respectivamente, y la tasa del esquema de codificación es $k/n = 1/2$.

Para mejorar el valor de esta tasa de código se aplica la selección de los datos de redundancia provenientes de los codificadores, de manera que por ejemplo, y como es muy comúnmente usado, si se elimina alternativamente una u otra de las salidas de paridad de cada codificador CCSR del esquema, la tasa del mismo pasa a ser de $k/n = 1/2$. No se aplica selección a los bits de información por que esto reduce la tasa del esquema.

Dos de las partes más importantes en el esquema de un codificador turbo son el permutador de datos, en particular su dimensión o longitud, y la naturaleza recursiva

(empleo de máquinas secuenciales de estados finitos de respuesta impulsiva infinita) de los codificadores componentes del esquema. El excelente funcionamiento de estos códigos es función de su naturaleza pseudo-aleatoria y la longitud del permutador. Por otra parte esta longitud no incrementa demasiado la complejidad de la decodificación, pero si agrega un retardo, que en algunas aplicaciones puede ser una desventaja.

2.6.2. El decodificador turbo

La esencia de la decodificación turbo es la iteración de información entre los decodificadores, que se asiste de la independencia estadística de las dos secuencias que han sido generadas por cada bit de información. La estructura del decodificador es como la que se ve en la figura 2.10 En el proceso de decodificación cada decodifi-

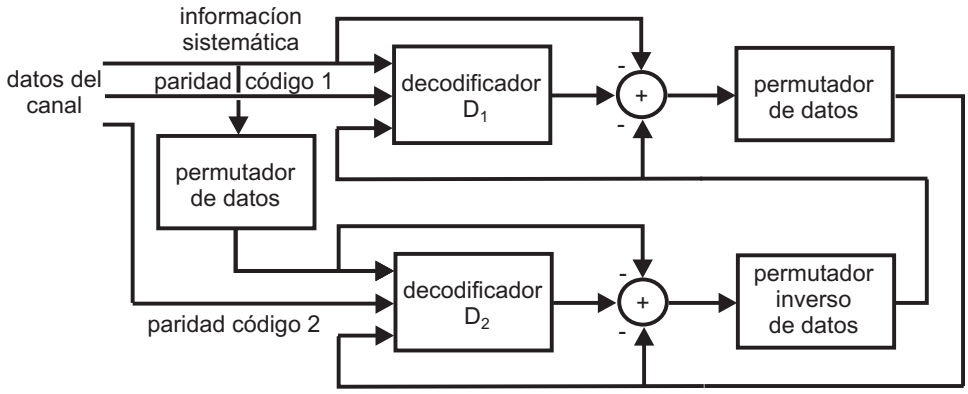

Figura 2.10: Esquema del decodificador turbo.

cador emplea información provista desde el canal, relativa al dato sistemático y de paridad correspondiente, junto con la información a priori proporcionada por el otro decodificador.

Sin embargo el decodificador no realiza una decisión rígida como en el caso del decodificador Viterbi tradicional, sino que entrega una estimación acerca de la información decodificada. Esta información da una estimación de la posibilidad de que el dato decodificado sea un '1' o un '0'.

Esta información se suele medir en forma logarítmica, a través de lo que se conoce como la relación logarítmica de probabilidad. Esta medición es muy adecuada por que resulta ser un número con signo, de manera que su polaridad indica si el bit es '1' (positiva) o '0' (negativa), mientras que su magnitud da una idea de cuan probable es el evento medido. Existen varios algoritmos que operan resolviendo estimaciones y entregando como salida también estimaciones [45, 46], entre los cuales se encuentra el algoritmo BCJR [42].

2.7. Algoritmo de encriptamiento simétrico estándar AES

El cifrador Rijndael [47, 48] resultó ser el finalista elegido entre cinco candidatos, para la implementación del algoritmo de encriptamiento simétrico estándar, conocido en inglés como *Advanced Encryption Standard* (AES). El AES se usa para la protección de información sensible, en reemplazo del DES (*Data Encryption Standard*), a partir del 26 de mayo del 2002.

Se trata de un cifrador que encripta [49] bloques de 128, 192 o 256 bits usando claves de 128, 192 o 256 bits. AES opera en un arreglo de 4×4 bytes, llamado Matriz de Estado (algunas versiones de Rijndael con un tamaño de bloque mayor tienen columnas adicionales en la Matriz de Estado).

El proceso consiste en una serie de cuatro transformaciones matemáticas, la cuales se repiten 10, 12 o 14 veces, dependiendo de la longitud del bloque y de la longitud de la clave. Todos los ciclos, excepto el último son similares y consisten de las siguientes transformaciones:

- Transformación *ByteSub* (sustitución de bytes): en este paso se realiza una sustitución no lineal donde cada byte de la Matriz de Estado es reemplazado con otro de acuerdo a una tabla de búsqueda.
- Transformación *ShiftRow* (desplazamiento de filas): en este paso se realiza un transposición donde cada fila de la Matriz de Estado es rotada de manera cíclica un número determinado de veces.
- Transformación *MixColumn* (multiplicación de columnas): por efecto de esta transformación, cada columna de la Matriz de Estado se representa como un polinomio de grado 3 sobre $GF(2^3)$ y es multiplicado por un polinomio prefijado. La operación se realiza módulo otro polinomio, también fijo, $C(x) = x^4 + 1$.
- Transformación *AddRoundKey* (or-exclusiva): consiste en la adición sobre el campo finito GF (or-exclusiva) entre los bits de la Matriz de Estado y los bits de la clave.

En la figura 2.11 se indica la secuencia de las transformaciones que se le aplican al texto plano para obtener el texto encriptado. La primera transformación que se ejecuta es la *AddRoundKey* entre el texto plano y la llave. Luego se realiza uno de los procesamientos más importantes en el desarrollo del algoritmo el cual se lo denomina expansión de la clave.

Este proceso consiste en la generación de N claves distintas (normalmente $N = 10$) ya que las transformaciones *ByteSub*, *ShiftRow*, *MixColumn* y *AddRoundKey* se

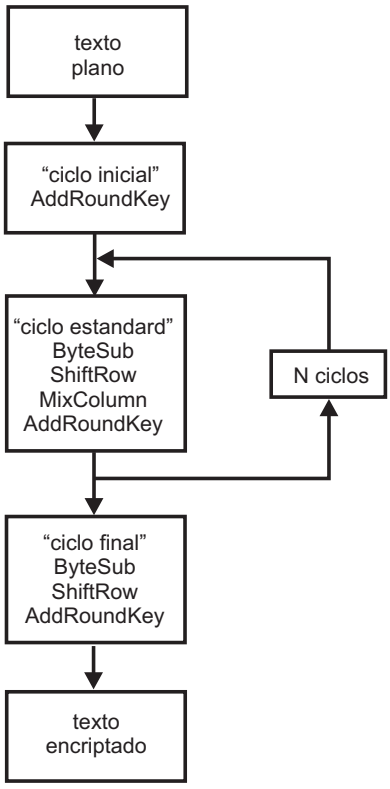

Figura 2.11: Proceso de encriptado.

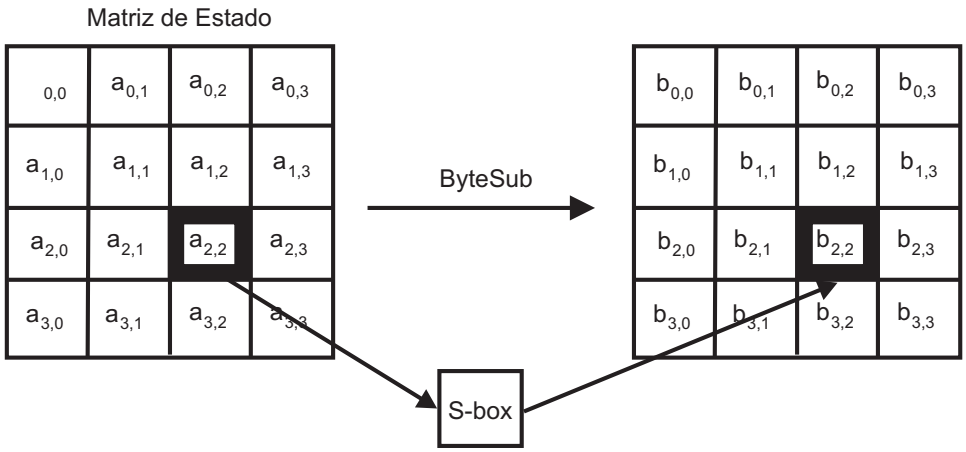

Figura 2.12: Transformación *ByteSub*.

repetirán N veces o rondas (en inglés *round*). De esta forma se genera una clave diferente para cada ronda. A continuación se describen brevemente cada una de las transformaciones.

2.7.1. *ByteSub*

La transformación *ByteSub*, es una sustitución no lineal, que opera independientemente en cada byte de la matriz. Cada byte del bloque de entrada es invertido sobre $GF(2^8)$ y luego sufre una transformación afín. La operación completa puede realizarse mediante una matriz de sustitución, conocida con el nombre de *S-Box*. La transformación *ByteSub* consiste en tomar un elemento (byte) $a_{i,j}$ de la Matriz de Estado de cuatro filas por cuatro columnas, y utilizarlo como subíndice necesario para extraer un valor de la *S-box*. Esta transformación se ilustra en la figura 2.12.

2.7.2. *ShiftRow*

En esta transformación los bytes de las tres últimas filas son desplazados cíclicamente en distintas cantidades o posiciones, denominadas en inglés *offsets*. En la primer fila el *offset* es nulo, es decir, que la fila no se desplaza. En la segunda fila, el desplazamiento es de 1 byte hacia la izquierda, en la tercera es de 2 bytes y en la última es de 3 bytes, también hacia la izquierda. Este proceso se puede apreciar en la figura 2.13.

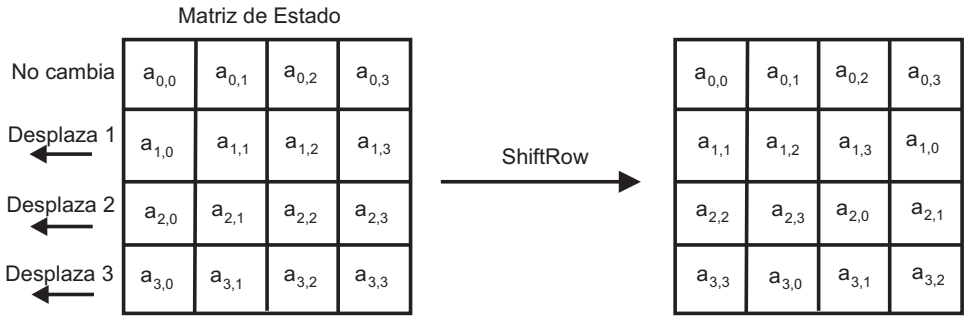

Figura 2.13: Transformación *ShiftRow*.

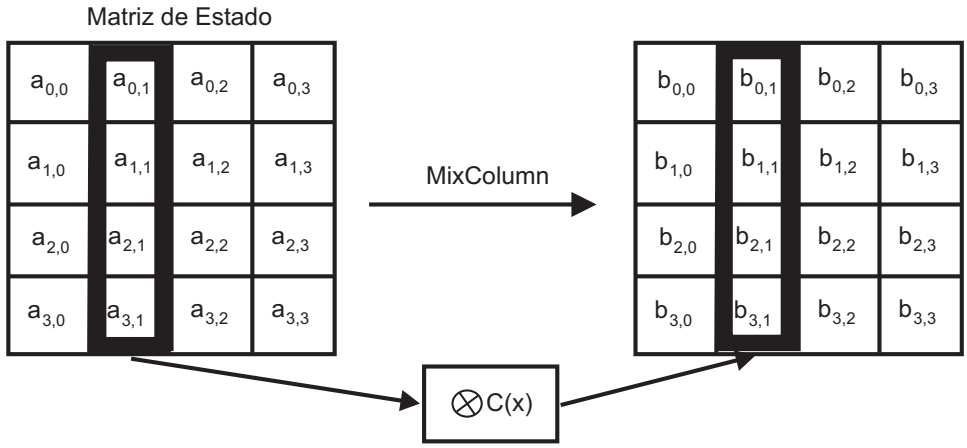

Figura 2.14: Transformación *MixColumn*.

2.7.3. *MixColumn*

La transformación *MixColumn* opera columna por columna de la Matriz de Estado. Cada columna se trata como un polinomio en el campo $GF(2^8)$ y luego se multiplica en operación módulo $x^4 + 1$ con un polinomio fijo $C(x)$. Este proceso se indica en la figura 2.14.

2.7.4. *AddRoundKey*

Como muestra la figura 2.15, en esta transformación la llave se combina con la Matriz de Estado. En cada ronda se obtiene una llave de la clave principal *key*. Cada llave es del mismo tamaño de la Matriz Estado. La llave se agrega combinando cada byte $a_{i,j}$ de la Matriz Estado con el correspondiente byte $k_{i,j}$ de la llave, usando la operación or-exclusiva, XOR.

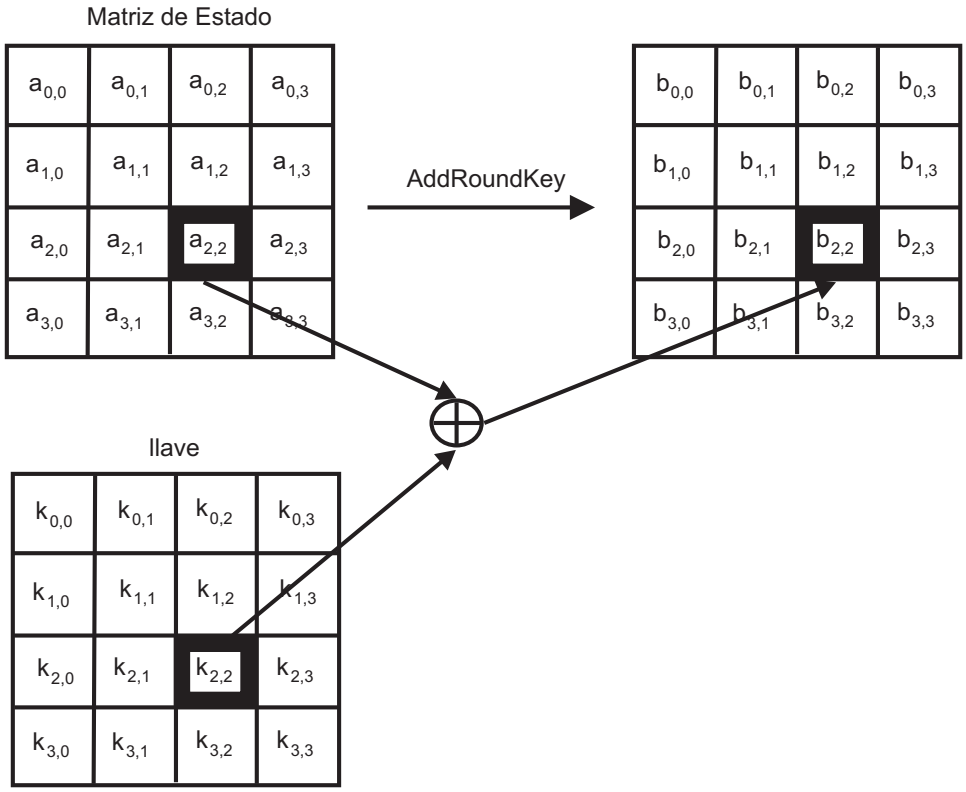

Figura 2.15: Transformación *AddRoundKey*.

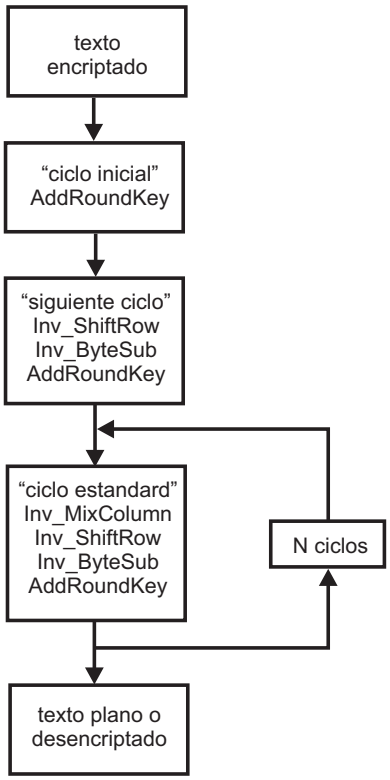

Figura 2.16: Proceso de desencriptado.

2.7.5. El proceso de desencriptado

El proceso de desencriptado se puede ver en la figura 2.16. La transformación *AddRoundKey* es igual que en el proceso de encriptado. Las demás transformaciones son la inversas de las transformaciones explicadas en el proceso de encriptado.

2.8. Utilización de la programación lineal para decodificar códigos LDPC

Resolver un problema de Programación Matemática (PM) es buscar el máximo (o el mínimo) de una función algebraica de variables ligadas por ecuaciones o inecuaciones algebraicas de cualquier grado llamadas restricciones. En el caso más simple, donde la función a maximizar (minimizar) y todas las restricciones son de primer grado, el problema recibe el nombre de Programación Lineal (PL) .

El gran interés suscitado por la PL se ha debido el desarrollo de la economía aplicada y de las técnicas modernas de gestión. Ahora bien, no es suficiente el saber resolver estos problemas sino que es preciso conocer el tiempo necesario para obtener la solución y el coste de esta obtención, ya que en muchos casos nos encontraremos con problemas con un gran número de variables y restricciones.

A partir de 1960 las áreas de aplicación se multiplican y la dimensión de los problemas cuya solución se hace posible aumenta rápidamente, suscitando un interés creciente por parte de los especialistas, organismos (públicos o privados) y empresas industriales. El progreso de la informática permite atacar problemas con varios miles de restricciones.

2.8.1. Formulación del problema de programación lineal

El objeto de la programación lineal es optimizar (minimizar o maximizar) una función lineal de n variables sujeto a restricciones lineales de igualdad o desigualdad. Más formalmente, se dice que un problema de programación lineal consiste en encontrar el óptimo (máximo o mínimo) de una función lineal en un conjunto que puede expresarse como la intersección de un número finito de hiperplanos y semiespacios en $\Re^n$.

En forma más general de un problema de programación lineal consiste en minimizar o maximizar:

$$Z = f(\mathbf{x}) = \sum_{j=1}^{n} c_j x_j \tag{2.83}$$

sujeto a:

$$\begin{aligned}
&\sum_{j=1}^{n} a_{ij}x_j = b_i, \quad i = 1, 2, ..., p-1 \\
&\sum_{j=1}^{n} a_{ij}x_j \geq b_i, \quad i = p, ..., q-1 \\
&\sum_{j=1}^{n} a_{ij}x_j \leq b_i, \quad i = q, ...m
\end{aligned} \tag{2.84}$$

donde p, q y m son enteros positivos tal que:

$$1 \leq p \leq q \leq n \tag{2.85}$$

Lo que distingue un problema de programación lineal de cualquier otro problema de optimización es que todas las funciones que intervienen son lineales. La función lineal que se tiene en la ecuación 2.83 se denomina función objetivo o función coste, y es la función que que se ha de optimizar. En la ecuación 2.84 se representan todas las posibles alternativas de las restricciones, dependiendo de los valores de p y q. En este trabajo solo se considerara la opción de minimizar la función objetivo. Las variables del problema $x_1, x_2, \ldots, x_n$ se denominan variables de decisión. Las constantes c_j coeficientes de coste.

- **Solución factible**: un punto $\mathbf{x} = (x_1, x_2, ..., x_n)$ que satisface todas las restricciones de la ecuación 2.84 se denomina solución factible. El conjunto de todas estas soluciones es la región de factibilidad.

- **Solución óptima**: un punto factible $\tilde{\mathbf{x}}$ tal que $f(\mathbf{x}) \geq f(\tilde{\mathbf{x}})$ para cualquier otro punto factible $\mathbf{x}$ se denomina una solución óptima del problema.

El objetivo de los problemas de optimización es encontrar un óptimo global. Sin embargo, las condiciones de optimalidad sólo garantizan, en general, óptimos locales, si éstos existen [50]. Sin embargo, los problemas lineales presentan propiedades que hacen posible garantizar el óptimo global:

- Si la región factible está acotada, el problema siempre tiene una solución (ésta es una condición suficiente pero no necesaria para que exista una solución).

- El óptimo de un problema de programación lineal es siempre un óptimo global.

- Si $\mathbf{x}$ y $\mathbf{y}$ son soluciones óptimas de un problema de programación lineal, entonces cualquier combinación (lineal) convexa de lo mismos también es una solución óptima. Obsérvese que las combinaciones convexas de puntos con el mismo valor de la función de coste presentan el mismo valor de la función de coste.

- La solución óptima se alcanza siempre, al menos, en un punto extremo de la región factible [50].

Para poner al problema de programación lineal en forma más compacta, se define la matriz $\mathbf{A}$ de coeficientes de las condiciones del problema como:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

Al vector $\mathbf{b^T} = [b_1, b_2, \cdots, b_m]$ se lo denomina vector de restricciones. Se denomina $\mathbf{x^T} = [x_1, x_2, \cdots, x_n]$ al vector factible que que satisface todas las condiciones. El conjunto $\mathbf{F}$ de todos lo vectores factibles constituye la región factible.

Entonces, el problema de programación lineal en forma más compacta queda:

$$\begin{aligned} min \quad & \mathbf{c^T} \cdot \mathbf{x} \\ s \cdot a \quad & \mathbf{Ax} \geq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{2.86}$$

La región factible entonces es:

$$\mathbf{F} = \{\mathbf{x} \in \Re^n : \mathbf{Ax} \geq \mathbf{b},\ \mathbf{x} \geq \mathbf{0}\} \tag{2.87}$$

También se puede transformar en la denominada forma estándar (sólo con condiciones de igualdad):

$$\begin{aligned} min \quad & \mathbf{c^T} \cdot \mathbf{x} \\ s \cdot a \quad & \mathbf{Ax} - \mathbf{y} = \mathbf{b} \\ & \mathbf{x}, \mathbf{y} \geq \mathbf{0} \end{aligned} \tag{2.88}$$

Donde el vector $\mathbf{y}$ se denomina de variables de holgura. Si las condiciones fuesen $\mathbf{Ax} \leq \mathbf{b}$ para llegar a la forma estándar se debe poner $\mathbf{Ax} + \mathbf{y} = \mathbf{b}$.

Si el problema se trata de maximizar una función objetivo, se puede transformar en uno de minimización teniendo en cuenta que:

$$maximizar \quad \mathbf{c^T} \cdot \mathbf{x} = -minimizar \quad \mathbf{c^T} \cdot \mathbf{x} \tag{2.89}$$

2.8.2. Politopos

Se llama hiperplano H de vector característico $\mathbf{a} \in \Re^n, \mathbf{a} \neq 0$, al conjunto $H = \{\mathbf{x} \in \Re^n : \mathbf{a}^T\mathbf{x} = \mathbf{c}\}$, con $\mathbf{c} \in \Re$. Es decir un hiperplano es el conjunto de soluciones de una ecuación lineal.

Dado un hiperplano H caracterizado por su ecuación $\mathbf{a}^T\mathbf{x} = \mathbf{c}$, se llaman semiespacios cerrados de borde H a los conjuntos:

$$H_+ = \{\mathbf{x} \in \Re^n : \mathbf{a}^T\mathbf{x} \geq \mathbf{c}\} \tag{2.90}$$

y

$$H_- = \{\mathbf{x} \in \Re^n : \mathbf{a}^T\mathbf{x} \leq \mathbf{c}\} \tag{2.91}$$

Se denomina *politopo* al conjunto formado por la intersección de un número finito de semiespacios cerrados. El conjunto $P = \{\mathbf{x} \in \Re^n : \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq 0\}$, de soluciones de un programa lineal es un politopo convexo. Dado que la ecuación:

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = b_1 \tag{2.92}$$

es equivalente al sistema de desigualdades:

$$\begin{aligned} a_1x_1 + a_2x_2 + \cdots + a_nx_n &\leq b_1 \\ a_1x_1 + a_2x_2 + \cdots + a_nx_n &\geq b_1 \end{aligned} \tag{2.93}$$

es decir, resulta de la intersección de estos dos semiespacios cerrados, por lo que P es un politopo.

2.8.3. Decodificación de códigos de control de error usando PL

La decodificación con Máxima Probabilidad (MP) consiste en estimar la palabra de código que más se asemeje a la palabra $\mathbf{y}$ transmitida, cuando se recibe una palabra $\tilde{\mathbf{y}}$ proveniente de un canal con ruido.

La decodificación usando programación lineal consiste en encontrar una solución óptima, que cumpla con un sistema lineal de restricciones y satisfaga una función objetivo lineal [2, 51, 52, 53].

La decodificación en la técnica del control de error consiste en determinar, en función de la información que provee el vector recibido, cual de los vectores pertenecientes al código presenta mayor similitud con respecto a dicho vector. En este sentido este proceso puede ser interpretado como una operación de optimización. Esto justifica la utilización de la programación lineal para la decodificación de un código de control de error.

A continuación se detalla el procedimiento para decodificar códigos LDPC mediante el uso de la programación lineal, que se encuentra desarrollado en [2, 52].

Supongamos que se desea decodificar un código binario $C \subseteq \{0, 1\}^n$ que opera en un canal afectado con ruido Gaussiano aleatorio. Si $\mathbf{y} \in C$, representa a la palabra

transmitida e $\tilde{\mathbf{y}}$ a la palabra recibida, la función costo se define como:

$$\gamma_j = \ln\Big(\frac{Pr[\tilde{y_j}|y_j = 0]}{Pr[\tilde{y_j}|y_j = 1]}\Big) \tag{2.94}$$

Donde γ_j representa el costo de tener el bit y_j en 1, y la suma $\sum_j \gamma_j y_j$ el costo de tener la palabra de código $\mathbf{y}$. En base a esto, la palabra con mayor probabilidad de ser decodificada es la que posee el costo mínimo.

Para cada bit del código, se define una variable f_j con $j \in \{1 \cdots n\}$, entonces la solución óptima $\mathbf{f}$ que cumpla con la siguiente ecuación constituye una palabra del código:

$$minimizar \quad \sum_{j=1}^{n} \gamma_j f_j \quad s.a. \quad f \in C \tag{2.95}$$

Cada nodo paridad, que hay en un grafo bipartito (ver figura 2.8) define un código local, el código global corresponde a la intersección de todos los códigos locales. En terminología de programación lineal PL [51] cada nodo paridad define un politopo de palabra de código local, y un politopo global es la intersección de todos los politopos locales.

Para definir un politopo de palabra de código local, se considera al conjunto de nodos símbolos d_j, con $j \in N(i)$, donde $N(i)$ representa el conjunto de subíndices de todos los nodos símbolos d_j que participan del nodo paridad h_i. El interés está centralizado en el subconjunto $S \subseteq N(i)$ que contiene un número par de nodos variables. Cada uno de estos subconjuntos corresponden a una palabra de código local, definida colocando $y_j = 1$ para cada índice $j \in S$, $y_j = 0$ para cada $j \in N(i)$ pero $i \ni S$, y colocando los otros y_j arbitrariamente.

A continuación mediante un ejemplo se verá como se presenta el problema de programación lineal para ser usado en la decodificación de códigos de control de error.

2.8.4. Ejemplo

Supongamos que se tiene un código caracterizado por una matriz $\mathbf{H}$:

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

De esta matriz se desprenden las siguientes ecuaciones de paridad:

$$\begin{aligned} y_1 \oplus y_2 \oplus y_3 &= 0 \\ y_2 \oplus y_3 \oplus y_4 &= 0 \end{aligned}$$

con $y_1, y_2, y_3, y_4 \in \{0, 1\}$. Si se llama N_i al número de unos de la fila i de la matriz $\mathbf{H}$, se tiene que: $N_1 = N_2 = 3$. Analizando la primera ecuación de paridad, se ve que debe cumplir:

$$\begin{aligned} y_1 &= y_2 \oplus y_3 \\ y_2 &= y_1 \oplus y_3 \\ y_3 &= y_1 \oplus y_2 \end{aligned}$$

y

$$y_1 + y_2 + y_3 = N_1 - 1$$

Se puede realizar un análisis similar para la segunda ecuación de paridad. Para resolver el problema mediante PL se define un politopo P con cuatro variables $\{f_1, f_2, f_3, f_4\}$, las cuales deben satisfacer las siguientes condiciones:

$$\begin{array}{ccc} (I) & (II) & (III) \\ f_1 \leq f_2 + f_3 & f_2 \leq f_3 + f_4 & 0 \leq f_1 \leq 1 \\ f_2 \leq f_1 + f_3 & f_3 \leq f_2 + f_4 & 0 \leq f_2 \leq 1 \\ f_3 \leq f_1 + f_2 & f_4 \leq f_2 + f_3 & 0 \leq f_3 \leq 1 \\ f_1 + f_2 + f_3 \leq 2 & f_2 + f_3 + f_4 \leq 2 & 0 \leq f_4 \leq 1 \end{array}$$

La condición (III) asegura que todos los f_j tomen valores entre cero y uno. Las condiciones (I) y (II), aseguran que el politopo P sea factible, es decir que el conjunto de palabras binarias de cuatro bits de longitud, que satisfacen las condiciones anteriores, son palabras del código C. La función costo a minimizar es $\sum_{j=1}^{n} \gamma_j f_j$, es decir:

$$minimizar \quad \gamma_1 f_1 + \gamma_2 f_2 + \gamma_3 f_3 + \gamma_4 f_4 \tag{2.96}$$

Si se desea llevar a un sistema del tipo $\mathbf{Ax} \leq \mathbf{b}$, las ecuaciones anteriores se deben modificar de la siguiente forma:

$$\begin{array}{ccc} f_1 - f_2 - f_3 \leq 0 & f_2 - f_3 - f_4 \leq 0 & 0 \leq f_1 \leq 1 \\ f_2 - f_1 - f_3 \leq 0 & f_3 - f_2 - f_4 \leq 0 & 0 \leq f_2 \leq 1 \\ f_3 - f_1 - f_2 \leq 0 & f_4 - f_2 - f_3 \leq 0 & 0 \leq f_3 \leq 1 \\ f_1 + f_2 + f_3 \leq 2 & f_2 + f_3 + f_4 \leq 2 & 0 \leq f_4 \leq 1 \end{array}$$

Para resolver el problema de programación lineal, se utiliza la función *LinProg* del programa Matlab®. Esta función resuelve el siguiente sistema de PL:

$$\begin{aligned} min_x \quad & \mathbf{c^T} \cdot \mathbf{x} \\ s \cdot a \cdot \quad & \mathbf{Ax} \leq \mathbf{b} \\ & l_{inf} \leq \mathbf{x} \leq l_{sup} \end{aligned} \tag{2.97}$$

Siguiendo con el ejemplo anterior, el sistema a ingresar adopta la forma:

$$\mathbf{c} = \begin{bmatrix} \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} 1 & -1 & -1 & 0 \\ -1 & 1 & -1 & 0 \\ -1 & -1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & -1 & -1 \\ 0 & -1 & 1 & -1 \\ 0 & -1 & -1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

$$\mathbf{b}^T = \begin{bmatrix} 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 \end{bmatrix}$$

con:

$$0 \leq f_j \leq 1 \tag{2.98}$$

2.9. Transmisión infrarroja

El incesante crecimiento de la demanda de computadoras personales y de sistemas de comunicaciones portátiles ha generado un gran interés en los enlaces inalámbricos de gran velocidad para ser usados en la interconexión de los elementos portátiles en las redes de área local (LAN) [54].

Estos enlaces deben ser compactos, consumir poca potencia y ser robustos contra el ruido ambiental y la interferencia producida por otros usuarios [55]. Como medio de transmisión en estos enlaces inalámbricos la luz infrarroja presenta algunas ventajas con respecto a la radio frecuencia [56], tales como un gran ancho de banda sin regulación, y la ausencia de interferencia entre enlaces que operan en diferentes habitaciones o separados por paneles opacos [57].

Sin embargo, los enlaces infrarrojos deben operar en la presencia de una fuerte radiación ambiental producida por la luz solar, la iluminación incandescente y fluorescente. También el rango de operación del enlace [58] queda restringido debido a que la potencia de transmisión debe limitarse para evitar un consumo elevado y para prevenir daños en la vista. Por estos motivos la relación señal a ruido (SNR) en el receptor pueda variar considerablemente con el consiguiente aumento de los errores producidos durante la transmisión [59, 60].

2.9.1. Modelo simplificado de un enlace infrarrojo

Para lograr una alta relación señal a ruido, en inglés *Signal to Noise Ratio* (SNR) los sistemas infrarrojos emplean modulación de la intensidad con detección directa, inglés *Intensity-Modulation Direct-Detection* (IM/DD)[61]. En modulación de intensidad, la señal transmitida X(t) es la potencia óptica instantánea. La señal recibida Y(t) es la corriente producida en el fotodetector del receptor, la cual es el producto de la respuesta R $(A.cm^2/W)$ del fotodetector y la potencia Pr recibida en su superficie. La potencia recibida está dada por [62, 63, 64, 65]:

$$P_r = Ho(\rho)\, P_t \tag{2.99}$$

donde $Ho(\rho)$ (cm^{-2}) es la ganancia óptica del enlace entre el transmisor y el receptor separados por una distancia ρ y P_t es la potencia promedio transmitida, la cual esta definida como:

$$P_t = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} X(t)\, dt \tag{2.100}$$

El ruido N_0 (A^2 /Hz) presente en el enlace, producido por la luz ambiental, se puede considerar como ruido blanco Gaussiano [66]. N_0 se supone constante e independiente

de la posición del receptor. Si la velocidad de transmisión del canal es r_b, se puede definir el parámetro γ_b [67] como la relación señal a ruido (SNR) dada por:

$$\gamma_b = \frac{R^2 P_r^2}{r_b N_0} = \frac{R^2 H_0^2(\rho) P_t^2}{r_b N_0} \tag{2.101}$$

entonces la probabilidad de error en términos de γ_b durante la transmisión estará dada por:

$$p_{be} = Q(\sqrt{\gamma_b}) \tag{2.102}$$

Como se puede observar en la ecuación 2.101, la relación señal a ruido depende del cuadrado de la potencia promedio óptica recibida P_r, esto implica un límite en el máximo alcance del enlace ρ y obliga a que se deba transmitir con una potencia P_t relativamente alta [68].

Capítulo 3

Conceptos básicos de Lógica programable y lenguaje VHDL

3.1. La evolución del diseño hardware

La electrónica nació a comienzos del siglo XX, desde aquellos días se han inventado y construido una infinidad de dispositivos electrónicos. Se han realizado una inmensa variedad de máquinas a partir de la combinación de unos pocos componentes.

Un factor fundamental de los impresionantes resultados obtenido por la ingeniería electrónica ha sido la creciente estandarización de los componentes. Esto ha permitido construir sistemas cada vez más complejos basándose en la funcionalidad de las partes que lo componen. La aparición de los circuitos integrados aceleró vertiginosamente este proceso. A continuación se detalla brevemente su historia [69]:

- 1948: transistor.
- 1960: puerta lógica.
- 1962: circuito con 4 compuertas lógicas.
- 1964: circuito con funciones lógicas complejas, 5 a 20 compuertas lógicas.
- 1967: circuito MSI (*Medium Scale Integration*), 20 a 200 compuertas lógicas.
- 1972: circuito LSI (*Large Scale Integration*), 200 2000 compuertas lógicas.
- 1977: circuito VLSI (*Very Large Scale Integration*), más de 2000 compuertas lógicas.

En la actualidad conviven dos fenómenos antagónicos. Por un lado, la gran variedad de dispositivos electrónicos permite diseñar y construir casi cualquier sistema electrónico. Simultáneamente, el mercado impone a muchos productos electrónicos exigentes cotas de tamaño, operatividad, coste, consumo y fiabilidad, que difícilmente se pueden conseguir con componentes y circuitos integrados estándares.

Esta paradoja ha originado la aparición de fabricantes de semiconductores que ofrecen soluciones ASIC (*Application Specific Integrated Circuits*). Son circuitos integrados diseñados para responder a una determinada aplicación. No son componentes universales para un mercado gigante y global, sino circuitos integrados con una funcionalidad específica, a medida del cliente.

3.1.1. Lógica programable

Los ASIC permiten disponer de circuitos con excelentes características de área, velocidad, consumo y complejidad, acordes con las necesidades del mercado. Sin embargo, el costo del diseño (herramientas, laboratorios, mantenimiento, aprendizaje,

fabricación y verificación de las muestras, errores o cambios, etc.), hace que la solución ASIC sea viable económicamente sólo a partir de una determinada escala de producción.

El problema es como combinar las ventajas de un circuito integrado a medida con el bajo costo de los circuitos estándares. Dentro del área digital, la solución intermedia apareció con los dispositivos lógicos programables, en inglés *Programmable Logic Device* (PLD).

Un dispositivo lógico programable es un circuito integrado, formado por una matriz de puertas lógicas y flip-flops, que proporcionan una solución al diseño de forma análoga, a las soluciones producidas por suma de productos, productos de sumas y multiplexores.

La estructura básica de un PLD permite realizar cualquier tipo de circuito combinacional basándose en una matriz formada por compuertas AND, seguida de una matriz de compuertas OR. Tres son los tipos más extendidos de PLDs, la PROM, PAL, y la PLA [70].

- **PROM**, memoria de solo lectura, en inglés *Programmable Read Only Memory*. Este tipo de dispositivo se basa en la utilización de una matriz de compuertas AND fija, seguida de una matriz OR programable. La matriz programable esta formada por líneas distribuidas en filas y columnas en las cuales los puntos de cruce quedan fijos por unos diodos en serie con unos fusibles que son los encargados de aislar las uniones donde no se requiera la función lógica. La fase de programación se realiza haciendo circular una corriente capaz de fundir el fusible en aquellas uniones donde no se desee continuidad. Por otra parte, para cada combinación de las señales de entrada, el codificador activa una única fila y a su vez activa aquella columna a las que esta todavía unida a través del diodo.

- **PAL**, lógica de arreglo programable, en inglés *Programmable Array Logic*. Este tipo de dispositivo se basa en la utilización de una matriz de compuertas AND programable, seguida de una matriz OR fija.

- **PLA**, arreglo lógico programable, en inglés *Programmable Logic Array*. En este caso, ambas matrices, la de compuertas AND, y la de compuertas OR son programables. Esta estructura permite una mejor utilización de los recursos disponibles en el circuito integrado, de tal forma que se genera el mínimo numero de términos necesarios para obtener una función lógica.

Un paso gigante en la evolución de los dispositivos programables fue la aparición de las FPGA (*Field Programable Gate Array*). Una idea simple y revolucionaria: integrar en una sola pastilla los elementos necesarios para realizar circuitos digitales,

junto con los mecanismos para su interconexión, de forma tal que posteriormente el usuario lo pueda configurar para una determinada aplicación. Esto permitió obtener un circuito integrado que realice una determinada función específica en cuestión de horas.

Con la irrupción de estos dispositivos en el mercado a mediados de los 80´s [71, 72], el desarrollo incesante de las herramientas de diseño asistido por computadora, en inglés *Computer Aided Design* (CAD) y la utilización de lenguajes de descripción de *hardware* (HDL) [73, 74, 75]; se ha modificado el diseño del *hardware* digital evolucionando desde el diseño ascendente (*bottom-up*), en donde el diseñador construye el sistema desde los componentes elementales hacia el sistema final, hasta el diseño descendente (*top-down*), en donde se diseña el sistema desde su especificación funcional y a través de un proceso de síntesis se obtiene su implementación final.

Los dispositivos lógicos programables tienen la ventaja de una realización física inmediata y una reprogramabilidad virtualmente infinita. El costo para volúmenes de producción pequeños y medianos es también más beneficioso para esta alternativa, comparándolo con el alto costo de los prototipos ASIC.

La lógica programable se acomoda especialmente a los países en desarrollo por la baja inversión inicial que permite contar con un sistema completo de desarrollo, por la posibilidad de fabricar series reducidas e incluso unitarias, por la minimización de la diversidad de stock y por la reusabilidad de los componentes que permite cubrir con pocos productos el espectro completo de aplicaciones.

Estos dispositivos pueden ser programados y reprogramados por el diseñador en su laboratorio, esto hace que la verificación del *hardware* diseñado sea mucho más rápida y suceptible de ser corregido en caso de error, con un ciclo de iteraciones rápido, con la consecuente baja en el costo del diseño. Adicionalmente, con un diseño cuidadoso del sistema, las aplicaciones implementadas con estos dispositivos pueden ser actualizadas, luego de entregadas al cliente, permitiendo adaptaciones de viejos diseños a nuevos protocolos y especificaciones [76].

En la actualidad la capacidad de los dispositivos lógicos programables es tal que se pueden implementar sistemas de complejidad de varios millones de compuertas equivalentes, y trabajar a cientos de MHz [77].

Otra característica importante es la capacidad de implementar en forma rápida procesadores, junto con sus *buses*, periféricos y memorias en el mismo dispositivo programable.

Es también significativo el avance en la integración, dentro de la lógica programable, de módulos y bloques con características específicas, tales como bloques configurables a nivel de función, módulos analógicos, de gestión de múltiples relojes y tensiones de alimentación, módulos complejos de memoria, etc.

Estas ventajas hacen que la lógica programable pueda competir eficazmente con

las tecnologías de fabricación de circuitos integrados ASIC, compartiendo el mercado en algunos casos y superándolas en otros. Sin embargo las FPGA´s tienen algunas desventajas comparativas con respecto a las tecnologías mencionadas anteriormente tales como:

- Son más costosas en grandes volúmenes de producción.
- Usualmente los sistemas implementados son más lentos y ocupan mayor área que su equivalente en ASIC.

La evolución de estos dispositivos, el detalle de sus arquitecturas y las diferentes etapas de desarrollo de su tecnología puede consultarse en [78, 79]. En el apéndice B se muestra la descripción de la estructura de los dispositivos lógicos programables.

3.2. Lenguajes de descripción de circuitos (Hardware)

La complejidad de los sistemas electrónicos crece de forma constante. Este hecho, conocido por los diseñadores y usuarios, ha sido motivado, entre otros aspectos, por la mejora en los procesos de fabricación, la diversidad de aplicaciones y la variedad de componentes en el mercado. Sin embargo, estos aspectos podrían no haber supuesto una ventaja si los procesos de diseño no hubiesen mejorado de igual forma.

Además de la complejidad, el mercado de sistemas electrónicos evoluciona también en otros aspectos; tan importante como la calidad y el coste, es el tiempo de puesta en el mercado de un producto. En consecuencia, los tiempos de diseño tienden a reducirse mientras la complejidad de los diseños va aumentando.

Para afrontar estos problemas se está dedicando un gran esfuerzo a las metodologías de diseño. Una metodología de diseño define un conjunto de procedimientos, reglas y herramientas para optimizar el proceso de diseño de sistemas electrónicos. El criterio de optimización depende de la naturaleza del sistema, de su campo de aplicación y de su complejidad. Algunos criterios típicos pueden ser [80]:

- Tiempo y costo de desarrollo.
- Costo del producto final.
- Reutilización del diseño.
- Garantía de éxito al primer intento.

Un lenguaje de descripción de hardware puede ayudar a resolver los problemas previamente planteados. Estas son algunas de las ventajas de un HDL:

- Permite reducir el tiempo de diseño en los proyectos.
- Permite realizar una verificación continua de la funcionalidad y prestaciones del sistema.
- Hacen al sistema independiente de la tecnología destino y de los detalles de la implementación final.

El HDL es una herramienta formal para describir el comportamiento y la estructura de sistemas usando un esquema textual [81].

El diseñador puede así describir la operación del sistema, definiendo qué es lo que el sistema debe hacer (comportamiento), cómo debe hacerlo (algoritmo) y con qué hacerlo (flujo de datos y estructuras) [74, 75].

Este lenguaje permite describir actividades que ocurren en forma simultanea y en forma secuencial. Posibilita la construcción de estructuras jerárquicas y también modelizar el concepto "tiempo", parámetro fundamental para la descripción de sistemas electrónicos.

Existen muchos lenguajes para la descripción de circuitos digitales, pero el lenguaje de descripción de circuitos integrados de muy alta velocidad, en inglés *Very High Speed Integrated Circuit Hardware Description Language* (VHDL) es uno de los de mayor popularidad. La mayoría de los otros lenguajes suelen ser propios de una determinada herramienta o fabricante de circuitos integrados, por ello resultan a veces más óptimos pero no estándar. El lenguaje VHDL tiene también un amplio campo de aplicación, desde el modelado para simulación de circuitos, hasta la síntesis automática de circuitos.

3.2.1. El lenguaje VHDL

El VHDL es un lenguaje de descripción y modelado diseñado para describir la funcionalidad y la organización de sistemas digitales, circuitos digitales y componentes. Si bien el lenguaje VHDL fue concebido como un lenguaje para el modelado y simulación lógica dirigida por eventos de sistemas digitales, también se utiliza para la síntesis automática de circuitos. Es un lenguaje de amplia y flexible sintaxis, que permite el modelado preciso en distintos estilos del comportamiento de un sistema digital, y el desarrollo de modelos de simulación.

En la actualidad la síntesis a partir de VHDL constituye una de las principales aplicaciones. Las herramientas de síntesis basadas en éste lenguaje [82, 83, 84] permiten obtener ganancias importantes en la productividad del diseño.

3.2.2. Niveles de abstracción y VHDL

La primera manera de reducir el número de componentes con los que se trata en un sistema electrónico es tratarlo a mayor nivel de abstracción. Por ejemplo, en los circuitos digitales se consideran normalmente a nivel lógico, y no a nivel eléctrico.

La principal ventaja de trabajar a niveles de abstracción altos es la disminución del número de elementos con los que el diseñador tiene que tratar. Por otro lado, las descripciones de alto nivel permiten modelar el comportamiento del sistema independientemente de su estructura final.

Los lenguajes de descripción de *hardware*, y el VHDL en particular, posibilitan la descripción de sistemas digitales a diferentes niveles de abstracción. En el apéndice C se muestran las formas en las cuales el lenguaje VHDL permite describir un circuito.

Las metodologías de diseño pueden ser clasificadas en dos grupos dependiendo del nivel de abstracción de las descripciones de partida:

- **Ascendente**, en inglés *bottom-up*, donde el diseñador construye el sistema de componentes elementales, como compuertas lógicas y transistores. Éste método necesita de una fase previa donde el sistema ha sido dividido en bloques más pequeños.
- **Descendente**, en inglés *top-down*, en este caso, el diseñador construye el sistema partiendo de una descripción funcional, y a través de un proceso de síntesis, llega a la implementación final (entendiendo por síntesis a la conversión de una descripción de alto nivel a otra de nivel inferior).

La solución *top-down* es más adecuada para metodologías de diseño basadas en HDL, y está teniendo gran difusión gracias a las herramientas de diseño asistido por computadora, en inglés *Computer Aided Design* (CAD) de síntesis y simulación.

3.2.3. El proceso de diseño usando VHDL

Especificación

La primera fase estratégica en un proyecto es la etapa de especificación. En ella todas las características funcionales del sistema deben quedar claramente identificadas antes de adentrarse en el diseño. La experiencia muestra que la mayoría de los problemas durante la etapa de diseño son debidos a inconsistencias y poca precisión durante la etapa de especificación.

Simulación

Dado un conjunto de estímulos de entrada, es posible conocer la respuesta del sistema ante ellos mediante una herramienta de simulación. Así, es posible prever las prestaciones del sistema antes de que éste sea fabricado. El VHDL fue inicialmente diseñado con el propósito de simulación. Por lo tanto cualquier descripción escrita en este lenguaje puede ser simulada bajo el mismo entorno.

Síntesis

Como se explicó previamente, la síntesis consiste en la transformación de una descripción de alto nivel en otra de menor nivel de abstracción. Dependiendo del nivel de abstracción que se parta, se pueden considerar dos clases de síntesis:

- **Síntesis de alto nivel**, que transforma una descripción algorítmica en un modelo a nivel transferencia de registro.

- **Síntesis de bajo nivel**, convierte una descripción a nivel de transferencia de registros en un conjunto de compuertas interconectadas.

Las herramientas de síntesis tienen algunas limitaciones en el tipo de código VHDL que aceptan. Por ejemplo, en las descripciones de síntesis de bajo nivel para síntesis, el reloj y las señales de inicialización deben estar claramente identificadas. De igual manera, los tipos de datos están restringidos, y los tipos reales no son aceptados normalmente.

Otro aspecto importante es la dependencia de los resultados del estilo de descripción. Si el código VHDL, no es descrito de acuerdo con el punto de vista del sintetizador, los resultados en términos de área o velocidad pueden no ser los óptimos.

Test

El siguiente paso en el proceso de diseño es la especificación del procedimiento de test que será utilizado para verificar las prestaciones del sistema. Cuanto más complejo sea el sistema, más compleja será la generación de los procedimientos de test.

3.2.4. Ventajas del VHDL

- El uso de VHDL permite la descripción de sistemas electrónicos digitales a cualquier nivel de abstracción.
- Al modelar al sistema independientemente de la tecnología, puede ser sintetizado y realizado mediante diferentes soluciones con muy poco esfuerzo. Esto también implica que el diseño sea reutilizable.
- El VHDL proporciona un interfaz común entre las diferentes personas involucradas en el proceso de diseño.
- El uso de herramientas de síntesis automáticas permite al diseñador concentrarse más en los aspectos de la arquitectura que en la estructura.
- La probabilidad de error se reduce considerablemente así como el tiempo de diseño y el esfuerzo dedicado. Todo ello redunda en una mejora de la calidad y tiempo de puesta en el mercado.

3.2.5. Inconvenientes del VHDL

- Como el diseñador pierde información precisa sobre la estructura del sistema, éste se hace más difícil de depurar, especialmente en aspectos críticos (tiempo, área y consumo).
- VHDL es un estándar no sometido a ninguna patente o marca registrada. No obstante es mantenido y documentado por el IEEE, existe una garantía de estabilidad y soporte [74].
- Algunas de las primitivas del lenguaje son difíciles de sintetizar o no tienen una correspondencia clara con el hardware [85].
- Son difíciles de implementar circuitos que realizan operaciones matemáticas complejas. Por ejemplo, los decodificadores del tipo LDPC y turbo códigos emplean una gran cantidad de operaciones de suma, resta, multiplicación y división de números en punto flotante.
- Los vendedores de herramientas de síntesis han definido subconjuntos para sus sintetizadores. Distintos sintetizadores admiten distintos subconjuntos y por ello se pierde en gran medida la ventaja de lenguaje estandarizado.

Capítulo 4

Implementaciones de sistemas de control de error para enlaces de baja complejidad

4.1. Introducción

En los sistemas de medida surge con frecuencia la necesidad de medir en puntos físicamente diferentes del lugar donde se va a procesar o presentar la información, se dice entonces que es necesario un sistema de telemetría. Cuando se deben registrar muchos datos, cuando el registro debe hacerse durante mucho tiempo, o simplemente cuando se desea mucha objetividad y fiabilidad en las medidas, se recurre a la automatización de éstas. Por tanto el propósito de un sistema de telemetría es obtener datos de una fuente situada en un lugar remoto o de difícil acceso y transmitirlos a un lugar donde puedan ser evaluados.

Generalmente un sistema de telemetría básico está compuesto por los siguientes componentes [86]: un sistema de recolección de datos, un módulo de codificación de la información a transmitir, un transmisor, un receptor, una unidad de decodificación y por último un sistema de procesamiento de los datos recibidos.

La recolección de datos se realiza mediante sensores o transductores que convierten una variable física en una señal eléctrica. Esta señal generalmente es muy pequeña, por tanto debe ser acondicionada antes de ser aplicada al transmisor. En el módulo de codificación se agregan bits adicionales a los datos, para así poder detectar o corregir posibles errores durante la transmisión. La unidad de decodificación es la encargada de detectar y corregir los posibles errores, de forma tal de restaurar los datos originales y entregarlos al sistema de procesamiento.

La tarea del transmisor y receptor es la de establecer el vínculo físico entre el lugar en que se realiza la medición y el sitio en que se la procesa. En este intercambio de datos, es de vital importancia el tamaño, la portabilidad y el costo del enlace empleado. Además estos enlaces deben ser compactos, consumir poca potencia y ser robustos contra el ruido ambiental y la interferencia producida por otros usuarios. Varias soluciones se han propuesto para lograr enlaces de corta distancia y alta velocidad.

El uso de una conexión eléctrica directa entre el elemento portátil y el fijo es la manera más simple de establecer el enlace. Esta conexión eléctrica es realizada mediante un cable y conectores en ambos extremos. El inconveniente con esta solución es que los conectores pueden ser costosos debido al tamaño pequeño de los elementos portátiles que unen, y además están sujetos a roturas debido a su uso repetido. El uso de enlaces de radio frecuencia permite solucionar las desventajas de las conexiones eléctricas fijas. Sin embargo, en esta solución hay varios factores que deben ser considerados como las regulaciones en la frecuencia y el ancho de banda del espectro permitido y las interferencias que se producen entre enlaces que operan próximos.

Otro medio de transmisión a ser considerado en los enlaces inalámbricos es el uso de luz infrarroja. El uso de radiación infrarroja puede ser una alternativa viable al uso

de radio frecuencia para las comunicaciones inalámbricas de corta distancia, debido a que el espectro de la región infrarroja no está regulado.

Como la radiación infrarroja queda confinada dentro de la habitación en la cual es generada, ésta no afecta a otros sistemas infrarrojos que estén operando en el exterior; así tampoco interfiere con el espectro de radio frecuencia [54, 67].

Sin embargo, los enlaces infrarrojos deben operar en la presencia de una fuerte radiación ambiental producida por la luz solar, la iluminación incandescente y fluorescente. También la potencia de transmisión se ve limitada debido al consumo de potencia y para evitar daños en los ojos, lo cual restringe el rango de operación del enlace. Debido a estos factores la relación señal a ruido (SNR) en el receptor puede variar considerablemente con el consiguiente aumento de los errores producidos durante la transmisión [87, 88, 89].

Para disminuir estos errores, se procede a codificar la señal a transmitir mediante el uso de códigos de control de error.

Dependiendo del tipo de aplicación requerida en la transmisión de datos, no siempre es necesario utilizar la técnica de control de error más poderosa o sofisticada.

Por tal motivo, en este capítulo se han implementados enlaces que utilizan diferentes estrategias de control de error, las cuales varían fundamentalmente en el grado de complejidad que presentan, y en la capacidad de control de error que poseen.

Primeramente se presenta un sistema de transmisión para uso médico basado en rayos infrarrojos. Éste es un sistema de telemetría completo, el cual utiliza un código Hamming para la detección y corrección de errores.

Luego es posible mejorar la performance del sistema modificando la estrategia de control de error utilizada. Por tal motivo se presenta la implementación de un codificador cíclico y por último el desarrollo de un codificador convolucional y decodificador Viterbi.

4.2. Sistema de transmisión para uso médico basado en rayos infrarrojos

En esta sección se presenta un prototipo de un sistema de telemetría basado en rayos infrarrojos que acoplado a sensores en forma de aguja o lanceta, permite medir diferentes parámetros fisiológicos. El sistema puede monitorear un máximo de 5 sondas remotas, cada una de las cuales es capaz de manejar hasta 15 parámetros diferentes entregados por los sensores correspondientes [90, 91, 92].

Para la recolección de los datos se utiliza multiplexación por división del tiempo [86]. El enlace físico entre la unidad encargada de realizar el monitoreo y las unidades remotas, se realiza mediante una transmisión óptica bidireccional con luz infrarroja directa [54, 93]. El sistema debe tener un consumo reducido debido a que opera con baterías. Para detectar y corregir posibles errores en los datos durante la comunicación se utiliza un módulo que realiza una codificación de Hamming [10].

El prototipo del sistema de codificación y decodificación fue realizado con Dispositivos Programables de ALTERA [1]. Para la comunicación infrarroja se utilizaron dispositivos comerciales, lográndose una distancia de transmisión promedio de 2 metros, manteniendo el consumo de corriente de la unidad transmisora infrarroja por debajo de los $500uA$.

4.2.1. Objetivo del sistema

El objetivo de este sistema de biotelemetría es poder medir en forma remota diferentes parámetros fisiológicos desde un sistema de monitorización colocado a unos metros de distancia. De este modo se puede tener permanentemente actualizado el estado de los mismos, además de poder realizar cualquier tipo de estudio estadístico con los datos almacenados [94].

El sistema está formado por 5 sondas y un circuito encargado de controlarlas, tal como se indica en la figura 4.1. Cada sonda está preparada para manejar hasta 15 parámetros fisiológicos diferentes.

El sistema monitor realiza las siguientes tareas: a)Interroga a las sondas en forma secuencial o en forma directa a una en especial, según se requiera; indicándole el o los parámetros que desee leer, mediante un comando de lectura. b)Envía una señal de comando para programar determinadas opciones en las sondas. Cuando el sistema monitor envía un comando, todas las sondas lo reciben, pero sólo lo contesta la sonda que ha sido seleccionada.

Para la prueba del prototipo de telemetría no se utilizaron las agujas sensoras sino que se simularon éstas mediante un sistema compuesto por un ordenador que simula los datos de las sondas y un multiplexor que los distribuye a las unidades de

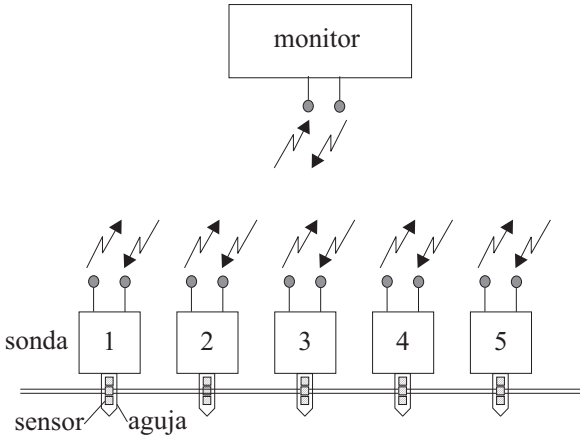

Figura 4.1: Sistema de telemetría.

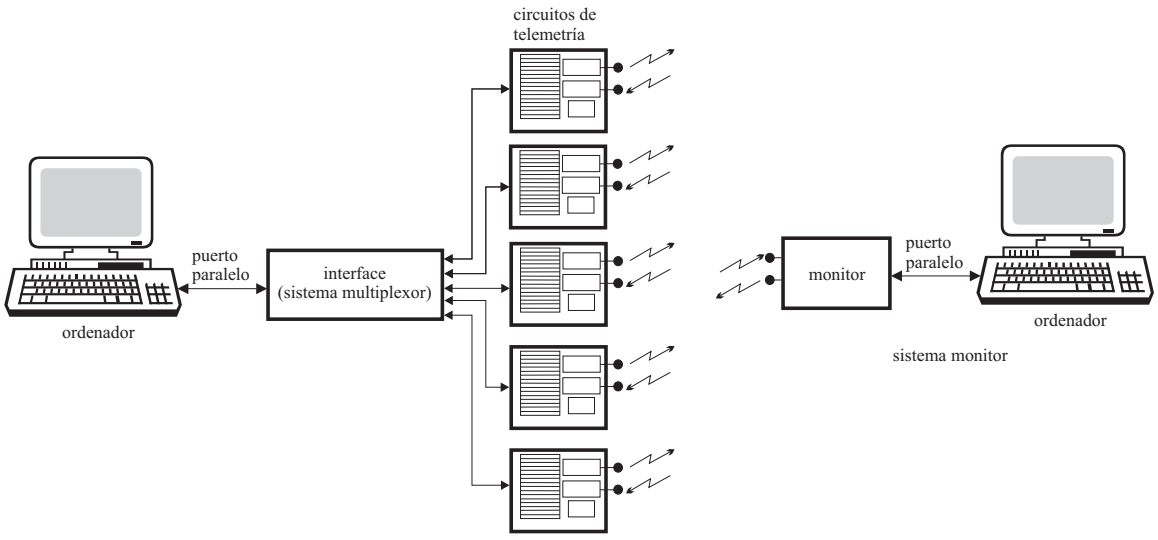

Figura 4.2: Esquema del sistema de evaluación propuesto.

telemetría, tal como indica la figura 4.2

En este caso, los circuitos de telemetría transmiten los datos que le suministra el primer ordenador a través del puerto paralelo, al sistema de monitorización, el cual los envía a un segundo ordenador que se encarga de procesarlos.

4.2.2. Descripción del sistema utilizado

Debido a las características impuestas por el sistema externo, el prototipo debe cumplir con las siguientes características:

- Tensión de alimentación del circuito infrarrojo: 2,9V .
- Corriente máxima de consumo durante la transmisión: 0,5mA.
- Frecuencia de reloj: 50KHz
- Número máximo de parámetro por sonda: 15.
- Bits por cada parámetro: 16.
- Número máximo de sondas: 5.
- Frecuencia máxima requerida para la transmisión para cada parámetro: 4 actualizaciones por segundo.
- Retraso máximo aceptable entre la medición de un parámetro y su transmisión: 1 segundo.
- Distancia de la transmisión sondas-monitor: 2 metros.
- Luz intensa incidiendo sobre las sondas.

4.2.3. Descripción de la sonda

La sonda, como muestra la figura 4.3, está compuesta por: a)Los sensores y sus correspondientes circuitos acondicionadores de señal. b)El circuito de telemetría, el cual además, contiene un banco de registros, en el que se almacenan los datos provenientes de los sensores. c)El transmisor y el receptor infrarrojo.

La sonda puede recibir dos tipos de comandos diferentes del sistema monitor: a)Un comando de petición de lectura, en el cual le indican que parámetros almacenados en los registros deben ser transmitidos. Como respuesta, la sonda envía la información solicitada. b)Un comando de programación, con el cual se pueden modificar determinadas opciones, (por ejemplo, una nueva dirección de la sonda, o que se produzca un

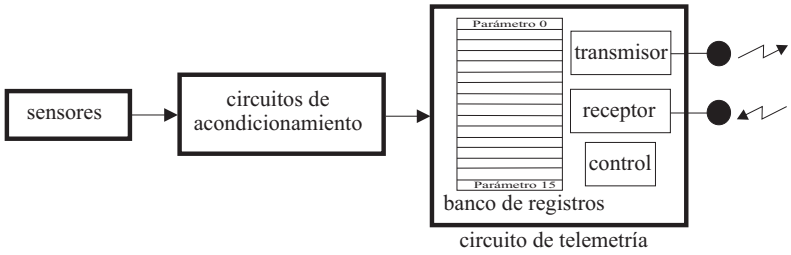

Figura 4.3: Sonda.

reset general), en este caso la sonda responde con un eco del comando recibido, como si se tratase del parámetro almacenado en el registro cero.

En la figura 4.4 se indican los cuatro bloques principales que componen el sistema de telemetría, estos son:

- Banco de registros: este bloque se encarga de almacenar los datos que le son suministrados por los sensores, Cuenta con 16 registros internos, de una longitud de 16 bits cada uno. En el registro cero se almacena el comando de programación que será retransmitido como una señal de eco y en los restantes se guardan los datos enviados por los sensores.

- Receptor: recibe el comando que envía el monitor, realiza la corrección o detección de errores utilizando el algoritmo de Hamming. Determina si el comando recibido es de petición de lectura o de programación; si el comando es de lectura indica al bloque de control cuales son los parámetros que se requieren; si el comando es de programación, deposita a este comando en el registro cero del banco de registros e indica al bloque de control que es necesario transmitir una señal de eco.

- Bloque de transmisión: está compuesto por un registro principal, en el cual se depositan los datos que son requeridos por el monitor, con el correspondiente protocolo de transmisión. En este bloque también se realiza la codificación Hamming de los datos a ser transmitidos.

- Bloque de control: se encarga de coordinar las tareas de los bloques descriptos anteriormente.

A continuación se detalla la función de cada uno de los terminales que posee el circuito:

- reset_sonda: produce un reset general en la sonda.

- clk50k: reloj externo de 50 KHz.

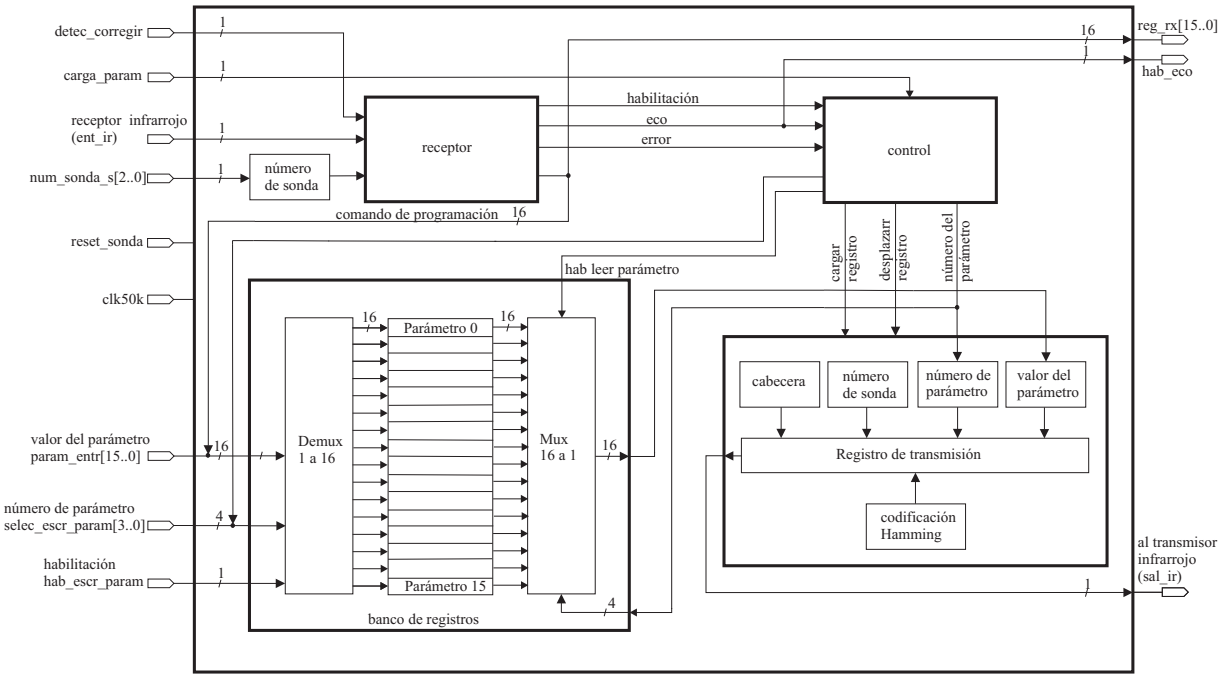

Figura 4.4: Circuito de telemetría de la sonda.

- num_sonda_s[2..0]: mediante una llave de selección se coloca la dirección de cada sonda.
- detec_corregir: colocando este terminal en cero, en el receptor se corrige un error por palabra; si se lo coloca en uno, se detectan uno o más errores.
- carga_param: se mantiene en uno durante toda la carga de los parámetros, evita que la sonda sea activada por el sistema monitor mientras se efectúa la recolección de datos.
- param_entr[15..0]: parámetro a ser almacenado.
- selec_escr_param[3..0]: indica la dirección del registro en el cual será almacenado el parámetro.
- hab_escr_param: al colocar esta entrada en uno se produce la carga del registro.
- hab_eco: indica al ordenador que simula los sensores, que se recibió un comando de programación.
- reg_rx[15..0]: envía al ordenador el contenido del comando de programación recibido.

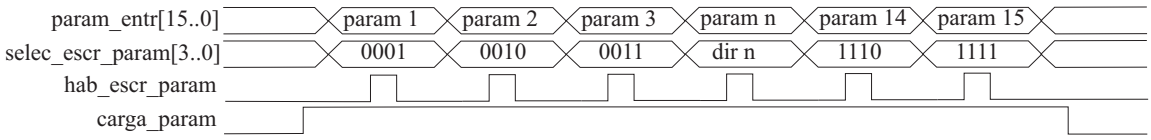

Figura 4.5: Carga de los parámetros.

- ent_ir: en este terminal ingresa la señal proveniente del receptor infrarrojo.
- sal_ir: envía la salida al transmisor infrarrojo.

En la figura 4.5 se indica el proceso de carga de los parámetros dentro de los registros del circuito de telemetría, el cual consta de las siguientes fases: a)Se coloca el terminal carga_param a uno mientras dure la carga de los registros, con esto se evita que la sonda responda a un requerimiento del monitor durante este proceso. b)En los terminales param_entr[15..0] se coloca el parámetro a cargar. c)En los terminales selec_escr_param[3..0] se indica la dirección del registro en el cual será almacenado el parámetro. d)Al colocar hab_escr_param igual a uno se produce la carga del registro. El proceso se repite hasta que se almacenan todos los parámetros.

4.2.4. Sistema monitor

El monitor es el encargado de recoger la información que se encuentra almacenada en los registros de la sondas. Puede interrogar a las sondas en forma secuencial o en caso de ser necesario a una sonda en particular; esto lo realiza enviando junto con el comando de lectura o programación el número de la sonda, cuyos parámetros son requeridos.

En la figura 4.6 se puede observar que el circuito monitor consta de dos bloques principales, un transmisor muy similar al utilizado en la sonda, y un receptor en el cual también se realiza la decodificación Hamming. La función de cada terminal es la siguiente:

- reset: produce un reset general del circuito monitor.
- clk: reloj externo de 2.5MHz.
- reg_entr[15..0]: comando que se desea enviar a la sonda.
- num_sonda[2..0]: sonda a la cual se envía el comando.
- lectura_prog: se coloca a cero si el comando es de petición de lectura o a uno si es de programación.

- hab_tx: comienza el proceso de transmisión.
- detec_corregir: colocando este terminal en cero, en el receptor se corrige un error por palabra; si se lo coloca en uno, se detectan uno o más errores.
- entr_ir: en este terminal ingresa la señal proveniente del receptor infrarrojo.
- fin_tx: indica al ordenado que se finalizó con el proceso de transmisión.
- transmitiendo: se mantiene en uno durante toda la transmisión.
- sal_ir: envía la salida al transmisor infrarrojo.
- parámetro_sal[15..0]: envía al ordenador los parámetros que se reciben de la sonda interrogada.
- num_param[3..0]; indica a que registro corresponde el parámetro que está en el bus anterior.
- hab_leer_param: indica al ordenador que puede leer el bus parámetro_sal[15..0] y el bus num_param[3..0].
- fin_recep: indica que la sonda envió todos los parámetros requeridos.
- error_byte: se coloca a uno si se detectó algún error durante la recepción.

Protocolo de transmisión

El sistema monitor es el responsable de controlar a las sondas, es decir, decide cuando y con que frecuencia las sondas deben ser contactadas. Para lograr esto el sistema emplea un protocolo basado en comandos [56], el monitor puede enviar dos tipos de comandos a la sonda, los cuales se ilustran en la figura 4.7a y 4.7b. Estos comandos son:

- **Comando de lectura:** especifica los parámetros que desea que le sean transmitidos por una determinada sonda, como respuesta, la sonda envía la información solicitada. Para lograr esto el comando de lectura utiliza una mascara de 15 bits, en la cual se coloca un uno en el valor que se quiere medir, por ejemplo, un uno en el bit 3 significa que la sonda debe enviar la lectura correspondiente al parámetro 3.

- **Comando de programación:** en este caso los 16 bits de datos transmitidos en este cuadro son enviados hacia la sonda para programar una determinada opción (por ejemplo una nueva dirección de la sonda). La sonda responde a este comando mediante un eco del dato transmitido en el comando. Este comando se envía como si

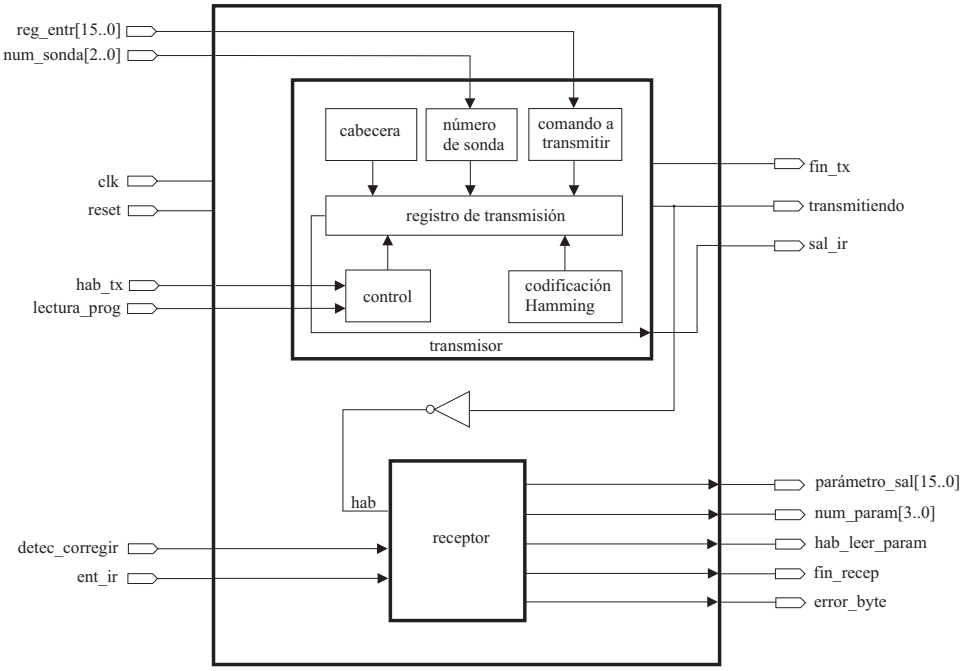

Figura 4.6: Sistema monitor.

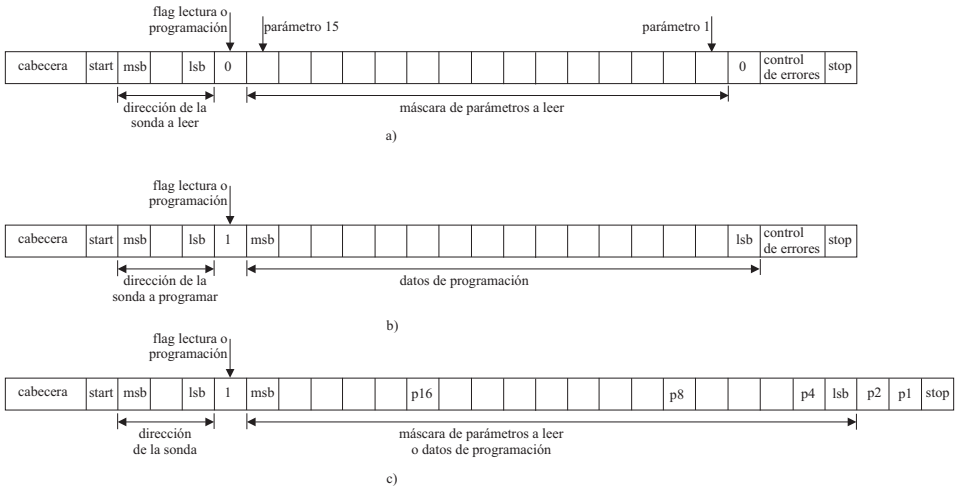

Figura 4.7: Comandos enviados por el monitor.

se tratase del parámetro 0 de la sonda. Si el comando de programación transmitido es “1111 1111 1111 1111”, éste produce un reset en el banco de registros en la sonda a la cual se envió el comando.

Si el circuito de recepción de la sonda detecta cualquier error en los cuadros de estos comandos no envía ninguna respuesta. Como se observa en la figura 4.7, los dos comandos cuentan con un cuadro compuesto por 5 campos:

- Cabecera, este campo se utiliza como señal de sincronismo, y consta de 54 bits.
- Dirección de la sonda a leer.
- Flag de lectura o programación, un cero indica que el comando es de lectura, y un uno que el comando es de programación.
- Máscara de parámetros a leer o datos de programación.
- Bits para el control de errores. En el prototipo propuesto, los bits para el control de errores (p16, p8, p4, p2 y p1) van intercalados con el campo de datos, tal como indica la figura 4.7c.

La respuesta de la sonda seleccionada consta de un cuadro compuesto por una serie de campos de longitud variable en función de la información solicitada por el monitor, como se ve en la figura 4.8a. Dichos campos son los siguientes:

- Cabecera, cumple la misma función que en el monitor.
- Dirección de la sonda seleccionada, con esto se evita que el resto de las sondas confundan a esta señal con un comando proveniente del monitor.
- Número de parámetro enviado.
- Valor del parámetro.
- Bits de control de errores.
- Cola, la cual indica que se terminó de transmitir todos los campos solicitados.

Los campos número de parámetros, valor del parámetro y control de errores, son de longitud variable, debido a que dependen de la cantidad de parámetros solicitados por el monitor. Al igual que en el monitor los bits de control de errores (p16, p8, p4, p2 y p1) van intercalados en el campo de parámetros, como indica la figura 4.8b.

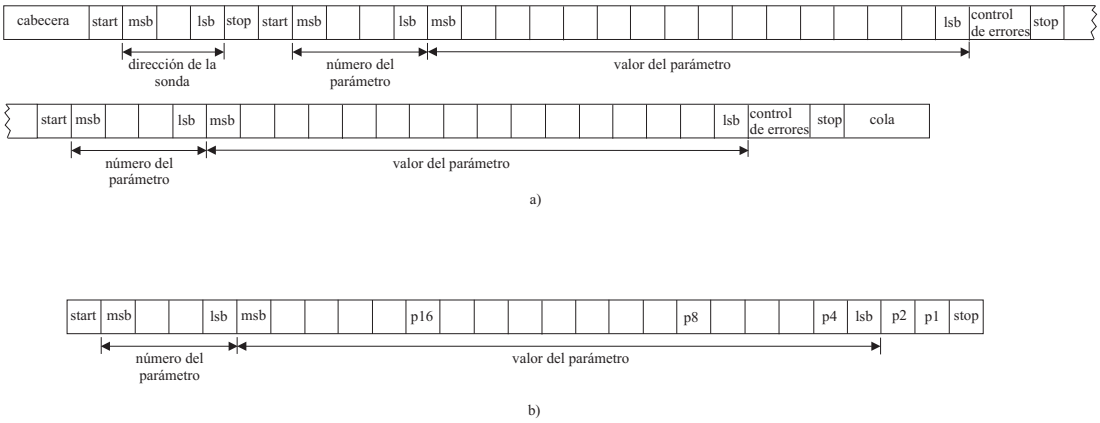

Figura 4.8: Respuesta de la sonda al monitor.

4.2.5. Sistema infrarrojo

Los componentes optoelectrónicos básicos requeridos para el enlace infrarrojo son el LED (*Light-Emitting Diode*) en el transmisor y el fotodetector en el receptor. La sensibilidad del fotodetector es función de la longitud de onda de la luz incidente, para maximizar la eficiencia, se debe elegir el LED para que la longitud de onda de su máxima emisión esté cerca del máximo de sensibilidad del fotodetector [95].

La distancia del enlace es función de la sensibilidad del receptor, de la potencia de salida y del ángulo de radiación del transmisor. Para ángulos pequeños la distancia del enlace está definida por la ecuación 4.1 [96, 97]:

$$\rho = \sqrt{\frac{(1000 \cdot P_t)}{s_{m\acute{i}n}}} \tag{4.1}$$

donde:

- ρ es la distancia en $[cm]$.
- P_t es la potencia transmitida en $[mW/Sr]$.
- $s_{m\acute{i}n}$ es la sensibilidad mínima del detector en $[\mu W/cm^2]$

La sensibilidad del receptor depende de la longitud de onda de la radiación recibida. La intensidad de potencia radiada está linealmente relacionada con la corriente que circula por el LED (I_{LED}) [98, 99], y se define como:

$$P_t = K_{LED} \cdot I_{LED} \quad [mW/Sr] \tag{4.2}$$

donde K_{LED} es la eficiencia óptica del LED medida en $[mW/(A \cdot Sr)]$

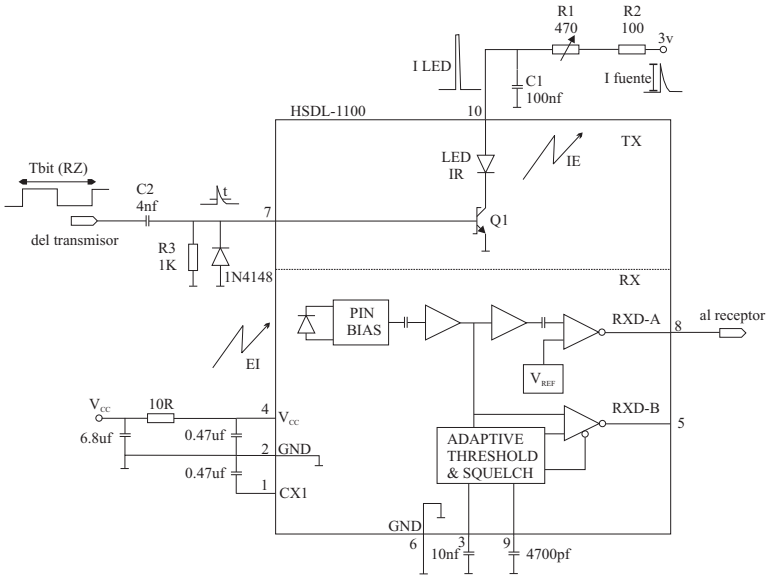

Figura 4.9: Transmisor y receptor infrarrojo.

En este prototipo por razones de tamaño y de orden práctico se empleó un transmisor-receptor integrado (transceptor). Se ensayaron varios modelos y se eligió el circuito HSDL-1100 de HEWLETT PACKARD [100], debido a que fue con el que se obtuvo la mayor distancia de transmisión. Se utilizó el sistema de modulación On Off Keying (OOK) [5]. Este dispositivo presenta una sensibilidad de $3,6\mu W/cm^2$ y una intensidad de potencia radiada de $177mW/Sr$, con lo que se obtiene un alcance de $221cm$. El inconveniente es que este valor de potencia radiada es para una corriente I_{LED} de $400mA$.

Según las especificaciones dadas, el consumo promedio durante la transmisión no puede superar los $0,5mA$, además debido al número de actualizaciones requeridas la velocidad debe ser de $12,5Kbits/seg$, por tanto se tienen 4 pulsos de reloj por bit. Suponiendo que sólo se transmite durante un cuarto de pulso, la corriente promedio durante la transmisión es:

$$I_{promedio} = I_{LED}/4 = 100mA \tag{4.3}$$

Como la información a transmitir es información binaria de 0's y 1's igualmente probables, la corriente promedio durante la transmisión se reduce a $50mA$. Este valor está muy por encima de los $0,5mA$ requeridos por las especificaciones, también debido al tipo de batería usado, la corriente pico del transmisor infrarrojo no puede superar los $8mA$. Para resolver este problema se recurrió a un circuito que acumula carga en una capacidad, como muestra la figura 4.9. Mediante el uso de un circuito derivador formado por C2 y R3 que se aplica en la base del transistor Q1, se disminuye el ancho

del pulso t durante el cual se entrega corriente al LED infrarrojo, con lo cual el ciclo de trabajo se reduce a:

$$ciclo\ de\ trabajo = \frac{t}{T_b} \tag{4.4}$$

En el capacitor C1 se almacena la carga mientras el transistor no conduce, esto permite disminuir el pico de corriente que entrega la batería de $3V$ cuando el LED es excitado.

Con este circuito se obtuvo un consumo promedio durante la transmisión de $0,34mA$, y los picos de corriente en la batería están por debajo de los $8mA$.

4.2.6. Descripción del codificador-decodificador empleado para la detección y corrección de errores

En este enlace se implementó un codificador-decodificador Hamming. En este código, para cualquier m, hay:

- $2^m - m - 1$ bits de información
- m bits de paridad
- $2^m - 1$ bits totales

Si se aplica el código Hamming a una palabra de 20 bits, se tienen 5 bits de paridad, dando una palabra total de 25 bits.

- $2^m - m - 1 = 20$ bits de información
- $m = 5$ bits de paridad
- $2^m - 1 = 31$ bits totales

Como de estos 31 bits, 6 siempre permanecen en cero, sólo se utilizan 25 bits. En el prototipo que se construyó, la palabra total tiene la siguiente distribución:

$$p_1\ p_2\ x_3\ p_4\ x_5\ x_6\ x_7\ p_8\ x_9\ x_{10}\ x_{11}\ x_{12}\ x_{13}\ x_{14}\ x_{15}\ p_{16}\ x_{17}\ x_{18}\ x_{19}\ x_{20} \tag{4.5}$$

donde los p_i son los bits de paridad y lo x_i son los bits de información.

En la figura 4.10 se muestra un esquema del proceso de codificación y decodificación de los bits de información. En el transmisor simplemente se realiza una asignación de los bits de información:

$$\begin{array}{lllll}
b_0 \rightarrow x_3 & b_1 \rightarrow x_5 & b_2 \rightarrow x_6 & b_3 \rightarrow x_7 & b_4 \rightarrow x_9 \\
b_5 \rightarrow x_{10} & b_6 \rightarrow x_{11} & b_7 \rightarrow x_{12} & b_8 \rightarrow x_{13} & b_9 \rightarrow x_{14} \\
b_{10} \rightarrow x_{15} & b_{11} \rightarrow x_{17} & b_{12} \rightarrow x_{18} & b_{13} \rightarrow x_{19} & b_{14} \rightarrow x_{20} \\
b_{15} \rightarrow x_{21} & b_{16} \rightarrow x_{22} & b_{17} \rightarrow x_{23} & b_{18} \rightarrow x_{24} & b_{19} \rightarrow x_{25}
\end{array}$$

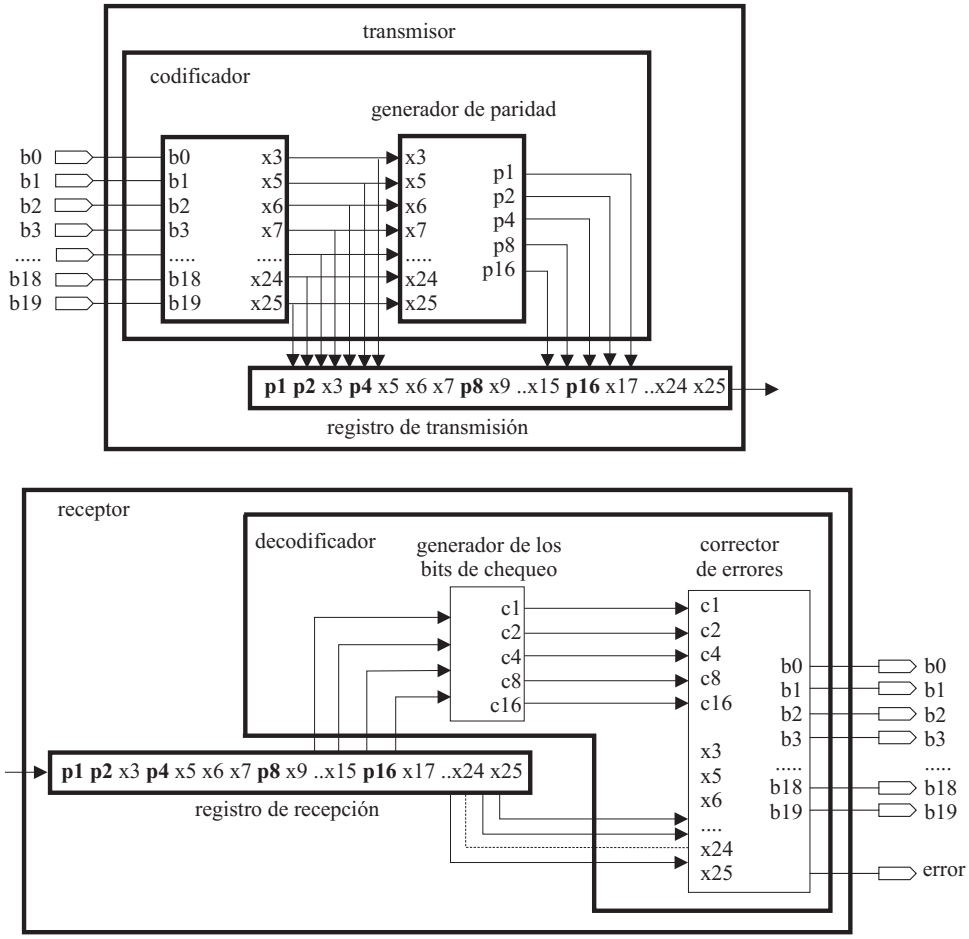

Figura 4.10: Codificador-decodificador Hamming.

Se define la operación de suma or-exclusiva de bits $\oplus(x_1, x_2, x_3)$ como $x_1 \oplus x_2 \oplus x_3$, de manera que en el generador de paridad se obtienen los bits de paridad como:

$$\begin{aligned} p_1 &= \oplus(x_3, x_5, x_7, x_9, x_{11}, x_{13}, x_{15}, x_{17}, x_{19}, x_{21}, x_{23}, x_{25}) \\ p_2 &= \oplus(x_3, x_6, x_7, x_{10}, x_{11}, x_{14}, x_{15}, x_{18}, x_{19}, x_{22}, x_{23}) \\ p_4 &= \oplus(x_5, x_6, x_7, x_{12}, x_{13}, x_{14}, x_{15}, x_{20}, x_{21}, x_{22}, x_{23}) \\ p_8 &= \oplus(x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{24}, x_{25}) \\ p_{16} &= \oplus(x_{17}, x_{18}, x_{19}, x_{20}, x_{21}, x_{22}, x_{23}, x_{24}, x_{25}) \end{aligned} \tag{4.6}$$

En el decodificador, con los bits que arriban se obtienen los bits de chequeo mediante las siguientes operaciones:

$$\begin{aligned} c_1 &= \oplus(p_1, x_3, x_5, x_7, x_9, x_{11}, x_{13}, x_{15}, x_{17}, x_{19}, x_{21}, x_{23}, x_{25}) \\ c_2 &= \oplus(p_2, x_3, x_6, x_7, x_{10}, x_{11}, x_{14}, x_{15}, x_{18}, x_{19}, x_{22}, x_{23}) \\ c_4 &= \oplus(p_4, x_5, x_6, x_7, x_{12}, x_{13}, x_{14}, x_{15}, x_{20}, x_{21}, x_{22}, x_{23}) \\ c_8 &= \oplus(p_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{24}, x_{25}) \\ c_{16} &= \oplus(p_{16}, x_{17}, x_{18}, x_{19}, x_{20}, x_{21}, x_{22}, x_{23}, x_{24}, x_{25}) \end{aligned} \tag{4.7}$$

Es decir los bits de chequeo se generan como:

$$c_i = p_{(generado\ en\ el\ receptor)} \oplus p_{(que\ arriba\ al\ receptor)} \tag{4.8}$$

Si el decodificador está configurado para corregir un error, con estos bits de chequeo se accede a la tabla de búsqueda 4.1 [3], en la cual se indica que bit se debe corregir, según el valor que tengan los bits de chequeo.

Si la combinación de los bits de chequeo, c_1, c_2, c_4, c_8 y c_{16} no concuerda con ninguno de los datos que están en la tabla de búsqueda, se coloca la salida 'error =1', indicando que se produjo en error que no se puede corregir. Si se está en la opción sin corrección de errores, se coloca la salida 'error =1' si cualquier bit de chequeo es diferente de cero.

4.2.7. Implementación

Para la realización del prototipo se utilizaron dos Dispositivos Lógicos Programables de ALTERA [1], uno para el monitor y otro para simular una de las sondas. El diseño de ambos circuitos se realizó en VHDL [84], para facilitar su portabilidad. Para el transmisor y receptor infrarrojo se usó el transceptor integrado de HEWLETT PACKARD HSDL-1100 [100] con las modificaciones ya expuestas para reducir el consumo de corriente durante la transmisión.

c_{16}	c_8	c_4	c_2	c_1	Bit a corregir
0	0	0	0	0	Sin error
0	0	0	1	1	x_3
0	0	1	0	1	x_5
0	0	1	1	0	x_6
0	0	1	1	1	x_7
0	1	0	0	1	x_9
0	1	0	1	0	x_{10}
0	1	0	1	1	x_{11}
0	1	1	0	0	x_{12}
0	1	1	0	1	x_{13}
0	1	1	1	0	x_{14}
0	1	1	1	1	x_{15}
1	0	0	0	1	x_{17}
1	0	0	1	0	x_{18}
1	0	0	1	1	x_{19}
1	0	1	0	0	x_{20}
1	0	1	0	1	x_{21}
1	0	1	1	0	x_{22}
1	0	1	1	1	x_{23}
1	1	0	0	0	x_{24}
1	1	0	0	1	x_{25}

Tabla 4.1: Bits a corregir, según la indicación de los bits de chequeo

Para medir la tasa de error, se situó a ambos dispositivos a una distancia de 2 metros, y se hizo que la sonda transmitiera secuencias de 100.000 palabras al monitor, en un ambiente iluminado con tubos fluorescestes e incidiendo alternadamente luz solar directa en el circuito transmisor y receptor infrarrojo. La tasa de error obtenida fue en promedio de 3/100.000, es decir 3 palabras de error cada 100.000 enviadas.

4.3. Implementación en lógica programable de un codificador cíclico

A continuación se expone la implementación de un codificador cíclico $(7, 4)$ implementado con dispositivos programables de ALTERA[1] que produce una interesante mejora en el funcionamiento de un enlace de transmisión [101, 102].

4.3.1. Codificador cíclico $(7, 4)$, implementado con dispositivos programables de Altera

En la figura 4.11 se ilustra el codificador cíclico $(7, 4)$ construido con dispositivos programables de ALTERA EPM7128S [1] usando el lenguaje de programación VHDL [84]. Este circuito está inspirado en el codificador presentado en la figura 2.3. Los terminales *coef1* y *coef2* permiten modificar los coeficientes del polinomio generador $g(X)$.

Se utiliza un registro realimentado de tres etapas para generar los tres bits de paridad que son agregados al vector de mensaje. Posee también un registro de siete etapas donde se almacena la palabra de código obtenida.

En la figura 4.12 se muestra el circuito utilizado para calcular el síndrome. Al igual que el codificador posee un registro realimentado de tres etapas, los cuales son los encargados de calcular el síndrome. Los valores utilizados en los terminales *coef1* y *coef2* para programar el polinomio generador $g(X)$ deben coincidir con los utilizados en el codificador. Como puede apreciarse, es muy sencillo modificar los dos circuitos presentados para poder utilizar cualquier código (n, k) cíclico.

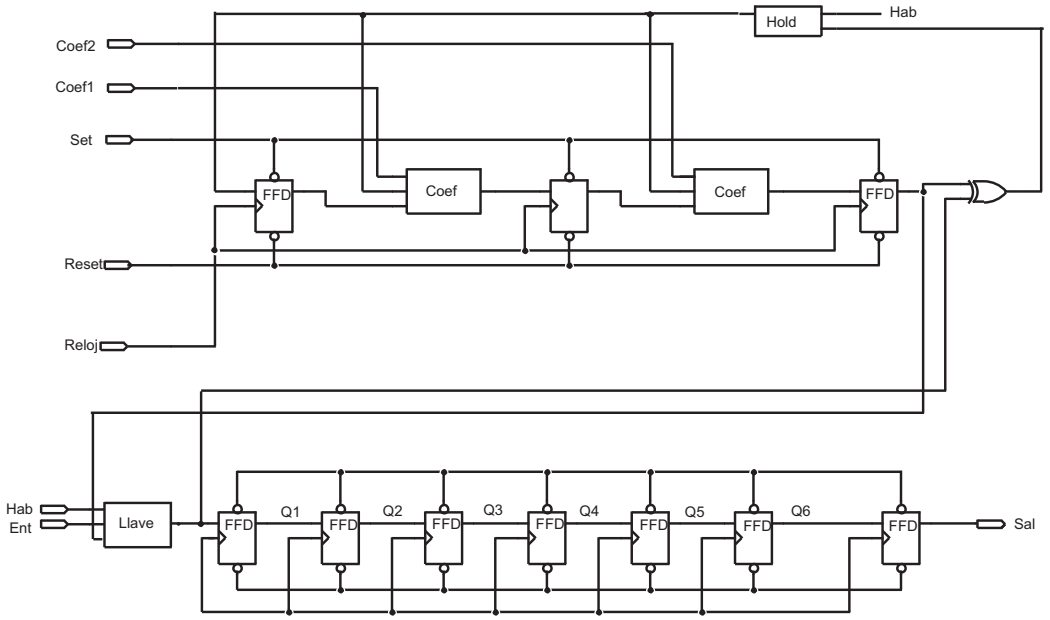

Figura 4.11: Circuito codificador cíclico $(7, 4)$ construido con dispositivos programables de ALTERA.

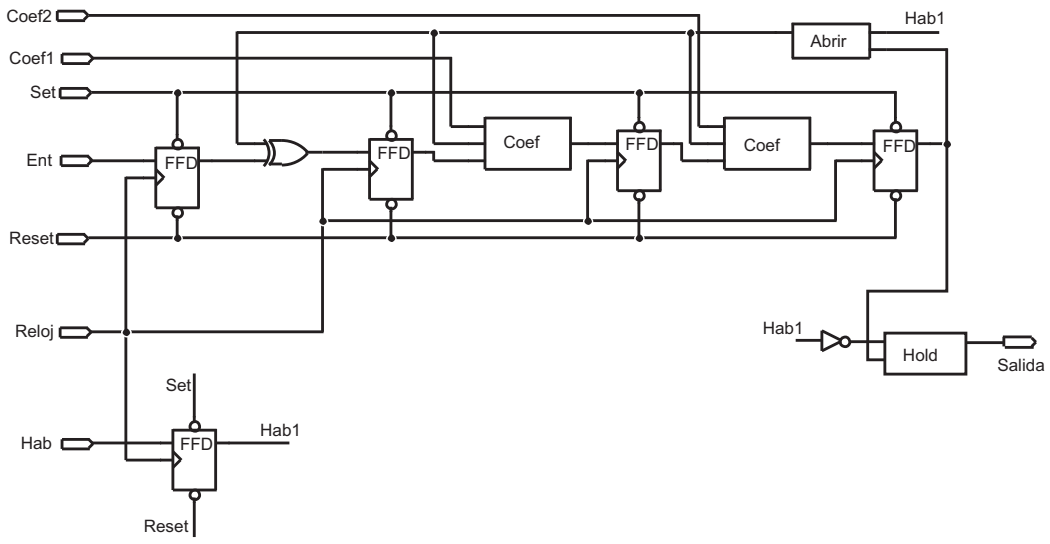

Figura 4.12: Circuito utilizado para calcular el síndrome.

4.4. Implementación con dispositivos programables de un codificador convolucional y decodificador Viterbi

En esta parte se expone un sistema implementado con dispositivos programables de ALTERA, para ser utilizado en comunicaciones infrarrojas interiores. Utiliza un codificador convolucional $(2, 1, 3)$ y un decodificador basado en el algoritmo de Viterbi. El codificador convolucional $(2, 1, 3)$, utiliza polinomios generadores $g1 = 101$ y $g2 = 111$ [3]. La palabra sin codificar tiene una longitud de 5 bits y la palabra codificada de 10 bits [103, 104].

4.4.1. Transmisor

El transmisor utilizado se muestra en la figura 4.13. El primer bloque conversor paralelo-serie simplemente transforma los datos de entrada de 5 bits, provenientes de un puerto paralelo de un ordenador, a forma serie. Luego estos bits en formato serie ingresan al bloque que realiza la codificación, el codificador convolucional. Este bloque opera de acuerdo a la información de codificación descripta en el diagrama de estado que se muestra en la figura 2.6. Cada nueva palabra comienza siempre en el estado 00. A la salida se obtiene una palabra codificada de 10 bits. A continuación se ingresa directamente al transmisor infrarrojo.

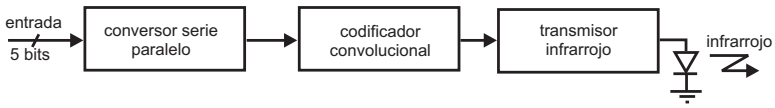

Figura 4.13: Sistema transmisor.

4.4.2. Receptor

En el sistema receptor se utiliza para decodificar la señal un decodificador Viterbi. Como se explicó anteriormente (sección 2.4.4), el fundamento de este algoritmo está en que no almacena todas las secuencias a las que da lugar el decodificador, sino, sólo el mejor camino desde el principio del diagrama de Trellis hasta cada nodo. El bloque principal del decodificador Viterbi se muestra en la figura 4.14, cada celda representa un nodo del diagrama de Trellis. El bloque tiene las siguientes entradas:

- **prog_estado=** se utiliza para programar la ubicación de cada celda dentro del diagrama de Trellis.

- **bA=** bit que provoca que del nodo o celda A se pase a la celda C.
- **regA=** contiene la secuencia de bits que son necesarios para llegar desde el nodo de partida al nodo A.
- **distA=** distancia de Hamming entre el vector recibido y la salida correspondiente del nodo A.
- **accumA=** contiene la suma de las distancias mínimas de Hamming, de los nodos que se recorrieron previamente antes de llegar al nodo A.

Las entradas **bB**, **regB**, **distB** y **accumB** tienen las mismas funciones que las anteriores. Las salidas de cada celda son:

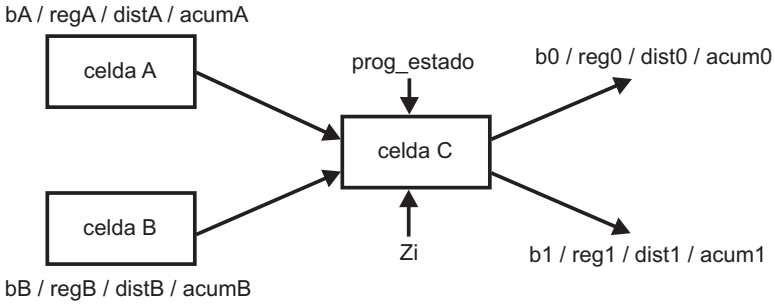

Figura 4.14: Celda básica del decodificador.

- **b0=** corresponde a la rama de salida que indica que el bit decodificado es un cero.
- **dist0=** distancia de Hamming entre el vector recibido y la salida correspondiente de este nodo cuando el bit de entrada a esta celda es un cero.
- **accum0=** contiene la suma de las distancias mínimas de Hamming, de los nodos que se recorrieron previamente antes de llegar a este nodo.
- **reg0=** contiene la secuencia de bits que son necesarios para llegar desde el nodo de partida a este nodo cuando el bit de entrada a esta celda es un cero.
- $\mathbf{Z}_i$= son los bits recibidos en ese nodo.

Las salidas **b1, dist1, accum1** y **reg1** tienen las mismas funciones que las anteriores, sólo que corresponden a cuando el bit decodificado es un uno.

En la figura 4.15, se muestra los diferentes bloques que componen cada nodo o celda del diagrama del Trellis. El circuito generador de estado produce la salida que

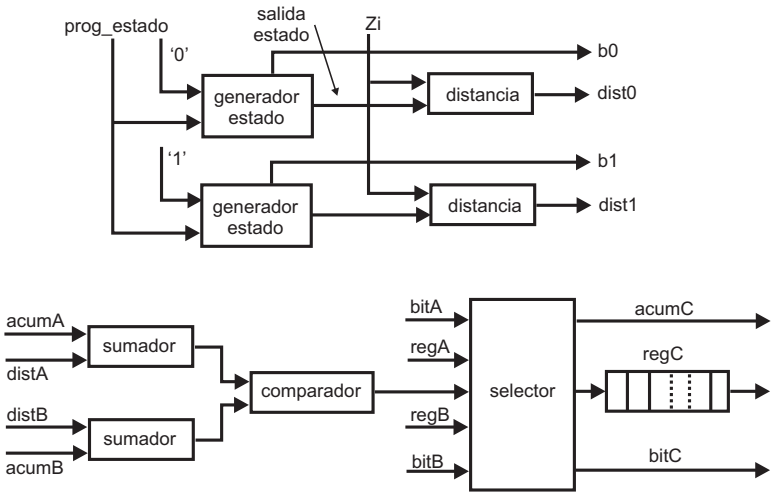

Figura 4.15: Bloques que componen cada nodo.

corresponde al estado que se programó, dependiendo si el bit de entrada es un cero o un uno. El bloque distancia, da la distancia de Hamming que hay entre la salida del generador de estado y los bits recibidos. Los bloques sumadores calculan la distancia total de cada camino que ingresa al nodo y mediante el comparador y selector se selecciona el camino que posee menor distancia de Hamming. Por último dependiendo de cual camino tiene menor peso se adiciona el bitA al regA o el bitB al regB, el cual pasará a ser el regC . La salida acumC es (acumA+distA) o (acumB+distB), dependiendo de cual tiene menor valor.

El transmisor se implementó en un dispositivo EPM7128SLC84-7 de la familia MAX 7000S [1], utilizando el 57 % de la capacidad máxima del dispositivo. El receptor utiliza 19 celdas como la mostrada en la figura 4.15, además de un sistema multiplexor para elegir el camino final con menor distancia de Hamming. Para su implementación se utilizó un dispositivo ALTERA EPF10K20RC240-A de la familia FLEX 10K, utilizando el 82 % de la capacidad máxima del dispositivo . El diseño del transmisor y el receptor se realizó en VHDL [84]. Como transmisor y receptor infrarrojo se utilizó el transceptor HEWLETT PACKARD HSDL-1100 [100].

4.4.3. Resultados

Para medir la tasa de error [11] en forma experimental, se construyó un prototipo utilizando dos plaquetas UP1 Educational Board, de ALTERA, y se situaron ambos dispositivos a una distancia variable entre 50cm y 250cm. Se hizo que el transmisor transmitiera secuencias de 600.000 palabras al receptor, en un ambiente iluminado con tubos fluorescestes e incidiendo alternadamente luz solar directa en el circuito transmisor y receptor infrarrojo. En la figura 4.16 se muestra como varía el porcentaje

de palabras recibidas con respecto al total de palabras transmitidas, para diferentes distancias, cuando se transmite una palabra sin codificar y una codificada. Se hace evidente la mejora que se obtiene con el uso de un codificador convolucional.

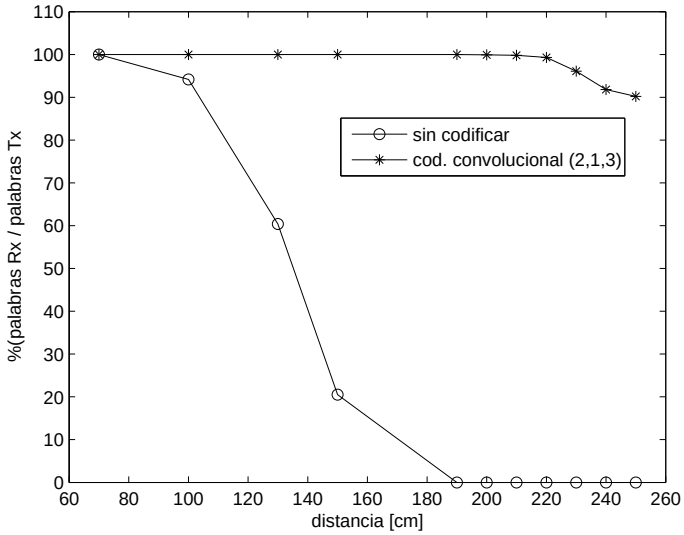

Figura 4.16: Porcentaje de palabras recibidas en función de la distancia del enlace.

4.5. Conclusiones

Primeramente se diseñó un sistema de telemetría completo, compuesto por varias sondas y un sistema monitor, que permite evaluar distintos parámetros biomédicos. Cada sonda puede almacenar hasta un máximo de 15 parámetros de 15 bits cada uno. El circuito monitor mantiene en todo momento el control del sistema, pudiendo decidir en cualquier momento cual es la sonda que se desea interrogar.

Como medio de comunicación del enlace se utilizó radiación infrarroja. Se mostró como es posible modificar la configuración del transmisor infrarrojo para disminuir notablemente su consumo de corriente.

Luego se expuso el diseño en lógica programable de un codificado-decodificador cíclico, que produce una mejora en el funcionamiento del enlace, su implementación es muy sencilla. También presenta la ventaja de permitir modificar fácilmente el sistema para poder utilizar cualquier código (n, k).

Por último, para aumentar la performance del sistema de telemetría se implementó un codificador convolucional, y un decodificador que utiliza el algoritmo de Viterbi.

Todas las implementaciones fueron realizadas utilizando dispositivos de lógica programable de ALTERA [1] y utilizando la herramienta de desarrollo MAX + PLUS II [84].

Se obtuvo una interesante mejora en el funcionamiento del enlace infrarrojo, permitiendo realizar enlaces de baja complejidad más confiables, capaces de operar en presencia de una fuerte radiación ambiental, manteniendo el consumo de potencia en niveles aceptables.

Capítulo 5

Implementaciones de sistemas de control de error para enlaces de alta complejidad

5.1. Introducción

Los códigos de paridad de baja densidad (LDPC) están recibiendo mucha atención debido a su funcionamiento cercano al límite de Shannon [6]. Un método que permite decodificar rápidamente códigos LDPC es el algoritmo de suma-producto propuesto por Gallager [17, 18, 19]. Los cocientes y productos en punto flotante involucrados en este algoritmo hacen que sea difícil su implementación en forma sencilla en lógica programable.

En este capítulo primero se prepone una forma de realizar la decodificación de códigos LDPC en forma iterativa utilizando sólo operaciones de suma-resta en punto fijo y dos tablas de búsqueda.

Con el algoritmo propuesto se implementa un decodificador que puede trabajar con cualquier tipo de matriz paridad (incluidas las generadas aleatoriamente, las generadas con una determinada regla de construcción, como las quasi-cíclicas, sean de tipo sistemático, o no sistemático), y se adapta en forma paramétrica cualquiera sea la tasa del código k/n.

Los resultados obtenidos muestran que no hay una gran diferencia en la decodificación al utilizar el método propuesto, con respecto al algoritmo clásico de suma-producto, lo cual permite implementar decodificadores en lógica programable de muy baja complejidad con muy buena performance.

Posteriormente se muestra como utilizando principalmente comparaciones y tablas de búsqueda es posible también implementar en forma sencilla un decodificación de códigos de control de error que utilice programación lineal.

5.2. Implementación de decodificadores LDPC en lógica programable, usando el algoritmo de suma-resta

En esta sección se presenta un algoritmo de decodificación LDPC de suma-resta en punto fijo, el cual permite fácilmente implementar un decodificador en lógica programable (FPGA)[105, 106, 107, 108, 109, 110, 111].

El decodificador implementado puede trabajar con cualquier tipo de matriz paridad (incluidas las generadas aleatoriamente, las generadas con una determinada regla de construcción, como las quasi-cíclicas, sean de tipo sistemático, o no sistemático), y se adapta en forma paramétrica cualquiera sea la tasa del código k/n. Esto le otorga una amplia versatilidad de aplicación y permite la utilización de los códigos LDPC más eficientes, que son los que utilizan matrices generadas aleatoriamente.

El algoritmo propuesto está basado en el algoritmo de suma-producto introducido por D. J. C. MacKay y R. M. Neal [18],

5.2.1. Algorítmo de suma-resta

Las bases del algoritmo de decodificación están descriptas en la sección 2.5.6. La simplificación que se propone permite que este algoritmo opere usando solo operaciones de suma, resta y búsqueda en tablas. Esta simplificación hace uso de una versión logarítmica de los cálculos desarrollados en el algoritmo original [18]. El método propuesto se basa en el supuesto de que si un número z es menor que uno, se lo puede representar como:

$$z = e^{-|wz|} \qquad con \qquad |wz| = |\ln(z)| \tag{5.1}$$

En el algoritmo propuesto, para dos números dados a y b que normalmente son intercambio de estimaciones entre los nodos de un grafo bipartito, las expresiones del tipo:

$$\ln(1 + e^{-||a|-|b||}) \qquad y \qquad \ln(1 - e^{-||a|-|b||}) \tag{5.2}$$

son aproximadas usando tablas de búsqueda llamadas $f_+(|a|, |b|)$ y $f_-(|a|, |b|)$ respectivamente [34, 45, 112]. Se utiliza como valor de entrada a ambas tablas la cantidad $||a| - |b||$.

El algoritmo que se presenta consta de la ejecución de los siguientes pasos:

Inicialización

El proceso de inicialización comienza ajustando los valores de las estimaciones Q_{ij}^a a lo valores de la probabilidad a priori de los símbolos f_j^a. La probabilidad a priori f_j^a

es la probabilidad de que el j ésimo símbolo sea a, con $a = \{0, 1\}$. En este caso, las variables Q_{ij}^0 y Q_{ij}^1 son inicializadas con los valores f_j^0 y f_j^1 respectivamente. Como f_j^a es un número menor que uno, es posible escribirlos como:

$$f_j^a = e^{-|wf_j^a|} \qquad con \qquad |wf_j^a| = |\ln(f_j^a)| \tag{5.3}$$

y

$$Q_{ij}^a = e^{|-wq_{ij}^a|} \qquad con \qquad |wq_{ij}^a| = |\ln(Q_{ij}^a)| \tag{5.4}$$

tomando en cuenta estas condiciones, sólo es necesario inicializar las variables $|wq_{ij}^0|$ y $|wq_{ij}^1|$ con los valores $|wf_{ij}^0|$ y $|wf_{ij}^1|$ respectivamente.

Para obtener $|wf_j^1|$ se parte de la definición de f_j^1 dada por la ecuación 2.60:

$$f_j^1 = e^{-|wf_j^1|} = \frac{1}{1 + e^{-\frac{2y_j}{\sigma^2}}} \tag{5.5}$$

obteniendo entonces:

$$|wf_j^1| = \ln(1 + e^{-\frac{2y_j}{\sigma^2}}) \tag{5.6}$$

y usando la tabla de búsqueda $f_+(|a|, |b|)$ definida en 5.2 queda:

$$|wf_j^1| = f_+(|\tfrac{2y_j}{\sigma^2}|, |0|) \tag{5.7}$$

De las ecuaciones 2.60, 2.61 se tiene:

$$\frac{f_j^0}{f_j^1} = e^{(-|wf_j^0| + |wf_j^1|)} = e^{-\frac{2y_j}{\sigma^2}} \tag{5.8}$$

entonces:

$$|wf_j^0| = \frac{2y_j}{\sigma^2} + |wf_j^1| \tag{5.9}$$

y usando la ecuación 5.7 queda:

$$|wf_j^0| = \frac{2y_j}{\sigma^2} + f_+(|\tfrac{2y_j}{\sigma^2}|, |0|) \tag{5.10}$$

Como se vio, la aproximación mediante el uso de tablas de búsqueda se definieron para expresiones del tipo dadas por las ecuaciones 5.2. Por tanto el uso de la tabla de búsqueda en las ecuaciones 5.7 y 5.10 es válido si y_j es positivo.

En la figura 5.1 se grafican $|wf_j^0|$ y $|wf_j^1|$ en función de $L = \frac{2y_j}{\sigma^2}$. Se puede observar que $|wf_j^1|$ con $y_j < 0$, es igual a $|wf_j^0|$ con $y_j > 0$. Entonces para poder utilizar la tabla de búsqueda $f_+(|a|, |b|)$ se deben usar, cuando y_j es negativo, las siguientes ecuaciones:

$$|wf_j^0| = f_+(|\tfrac{2y_j}{\sigma^2}|, |0|) \tag{5.11}$$

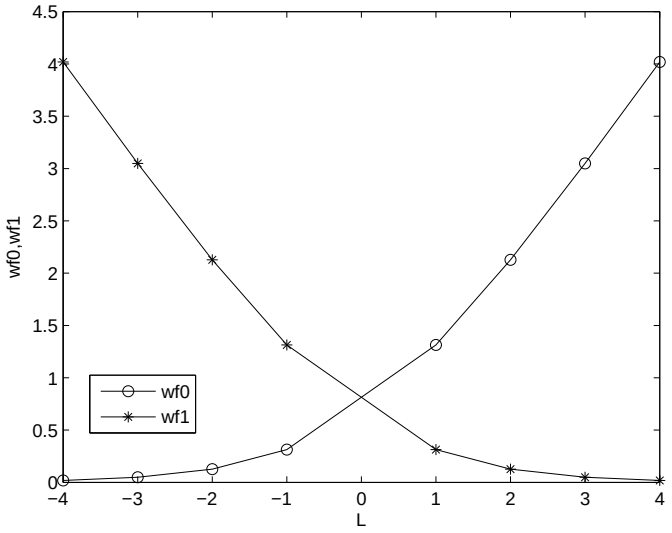

Figura 5.1: $|wf_j^0|$ y $|wf_j^1|$ en función de $L = \frac{2y_j}{\sigma^2}$.

y

$$|wf_j^1| = |\frac{2y_j}{\sigma^2}| + f_+(|\frac{2y_j}{\sigma^2}|, |0|) \tag{5.12}$$

en consecuencia, es necesario detectar el signo del símbolo y_j recibido, lo cual no es ningún inconveniente.

En la tabla 5.1 se presenta un resumen de como calcular $|wf_j^0|$ y $|wf_j^1|$:

$y_j > 0$	$\|wf_j^0\| = \frac{2y_j}{\sigma^2} + f_+(\|\frac{2y_j}{\sigma^2}\|, \|0\|)$	$\|wf_j^1\| = f_+(\|\frac{2y_j}{\sigma^2}\|, \|0\|)$
$y_j < 0$	$\|wf_j^0\| = f_+(\|\frac{2y_j}{\sigma^2}\|, \|0\|)$	$\|wf_j^1\| = \|\frac{2y_j}{\sigma^2}\| + f_+(\|\frac{2y_j}{\sigma^2}\|, \|0\|)$

Tabla 5.1: Cálculo $|wf_j^0|$ y $|wf_j^1|$

Paso horizontal

Se define δQ_{ij} como:

$$\delta Q_{ij} = Q_{ij}^0 - Q_{ij}^1 \tag{5.13}$$

debido a que $Q_{ij}^0 \leq 1$ y $Q_{ij}^1 \leq 1$, entonces también $\delta Q_{ij} \leq 1$, y se puede expresar como:

$$\delta Q_{ij} = e^{-|w\delta q_{ij}|} = e^{-|wq_{ij}^0|} - e^{-|wq_{ij}^1|} \tag{5.14}$$

reorganizando se tiene:

$$\delta Q_{ij} = (-1)^{s_{ij}} \cdot \left| e^{-|w\delta q_{ij}|} \right| \tag{5.15}$$

donde:

$$\left| e^{-|w\delta q_{ij}|} \right| = \left| e^{-|wq_{ij}^0|} - e^{-|wq_{ij}^1|} \right| \tag{5.16}$$

y

$$s_{ij} = 0 \qquad si \qquad |wq_{ij}^0| \leq |wq_{ij}^1| \tag{5.17}$$

o

$$s_{ij} = 1 \qquad si \qquad |wq_{ij}^0| > |wq_{ij}^1| \tag{5.18}$$

luego de las ecuaciones A.30, A.31, A.32 y en la tabla A.1 de la sección A, el valor $|w\delta q_{ij}| = \ln |\delta Q_{ij}|$ se puede aproximar como:

$$|w\delta q_{ij}| = \text{mín}(|wq_{ij}^a|) + |\ln(1 - e^{-||wq_{ij}^0| - |wq_{ij}^1||})| \tag{5.19}$$

donde se define a la función mín$(|w^a|)$, con $a = \{0, 1\}$ como:

$$\text{mín}(|w^a|) = |w^0| \; si \; |w^0| < |w^1| \quad \text{sino} \quad \text{mín}(|w^a|) = |w^1| \tag{5.20}$$

y utilizando una tabla de búsqueda, $|w\delta q_{ij}|$ puede ser escrito de la siguiente forma:

$$|w\delta q_{ij}| = \text{mín}(|wq_{ij}^a|) + f_-(|wq_{ij}^0|, |wq_{ij}^1|) \tag{5.21}$$

donde f_- es una tabla de búsqueda con entradas $\left| |wq_{ij}^0| - |wq_{ij}^1| \right|$.

Si $N(i)$ representa el conjunto de subíndices de todos los nodos símbolos $d(j)$ que participan del nodo de paridad $h(i)$, mientras que $N(i)\backslash j$ indica la exclusión del nodo j de ese conjunto, se calcula para cada i, j:

$$\delta R_{ij} = \prod_{j' \in N(i)\backslash j} \delta Q_{ij'} \tag{5.22}$$

y como esta productoria es menor o igual que uno, y utilizando la ecuación 5.15, se tiene:

$$\begin{aligned} \delta R_{ij} &= e^{-|w\delta r_{ij}|} \\ &= \prod_{j' \in N(i)\backslash j} (-1)^{s_{ij'}} |e^{-|w\delta q_{ij'}|}| \\ &= (-1)^{\sum s_{ij'}} \prod_{j' \in N(i)\backslash j} |e^{-|w\delta q_{ij'}|}| \end{aligned} \tag{5.23}$$

entonces es posible obtener:

$$|w\delta r_{ij}| = \ln |\delta R_{ij}| = \sum_{j' \in N(i)\backslash j} |w\delta q_{ij'}| \tag{5.24}$$

o como:

$$|w\delta r_{ij}| = \sum_{j \in N(i)} |w\delta q_{ij}| - |w\delta q_{ij}| \tag{5.25}$$

y

$$s\delta r_{ij} = \sum_{j' \in N(i) \setminus j} s_{ij'} \tag{5.26}$$

es decir:

$$s\delta r_{ij} = \sum_{j \in N(i)} s_{ij} \ - s_{ij} \tag{5.27}$$

entonces con las ecuaciones 5.25 y 5.27 se puede expresar la ecuación 5.23 como:

$$\delta R_{ij} = (-1)^{s\delta r_{ij}} |e^{-|w\delta r_{ij}|}| \tag{5.28}$$

con los valores de δR_{ij} anteriores, se calcula el conjunto:

$$R^0_{ij} = {}^1/_2 \cdot (1 + \delta R_{ij}) \tag{5.29}$$

$$R^1_{ij} = {}^1/_2 \cdot (1 - \delta R_{ij}) \tag{5.30}$$

usando la ecuación 5.28 se puede expresar la ecuación 5.29 en forma logarítmica como:

$$\ln(R^0_{ij}) = -|wr^0_{ij}| = \ln(1 + (-1)^{s\delta r_{ij}} \left|e^{-|w\delta r_{ij}|}\right|) - \ln(2) \tag{5.31}$$

si $s\delta r_{ij}$ es par, se tiene:

$$|wr^0_{ij}| = \ln(2) - \ln(1 + |e^{-|w\delta r_{ij}|}|) \tag{5.32}$$

y usando la ecuación A.15 de la sección A se tiene:

$$|wr^0_{ij}| = \ln(2) - f_+(|w\delta r_{ij}|, 0) \tag{5.33}$$

donde $f_+(|w\delta r_{ij}|, 0)$ es una tabla de búsqueda con entradas $|w\delta r_{ij}|$ y 0. Si $s\delta r_{ij}$ es impar, se tiene:

$$|wr^0_{ij}| = \ln(2) - \ln(1 - |e^{-|w\delta r_{ij}|}|) \tag{5.34}$$

como $(1 - |e^{-|w\delta r_{ij}|}|) < 1$ se puede obtener la siguiente expresión:

$$|wr^0_{ij}| = \ln(2) + |\ln(1 - |e^{-|w\delta r_{ij}|}|)| \tag{5.35}$$

y usando la ecuación A.31 de la sección A se obtiene:

$$|wr^0_{ij}| = \ln(2) + f_-(|w\delta r_{ij}|, 0) \tag{5.36}$$

donde $f_-(|w\delta r_{ij}|, 0)$ es una tabla de búsqueda con entradas $|w\delta r_{ij}|$ y 0.

Usando las ecuaciones 5.28 y 5.30 se puede expresar:

$$\ln(R_{ij}^1) = -|wr_{ij}^1| = \ln(1 - (-1)^{s\delta r_{ij}} \left|e^{-|w\delta r_{ij}|}\right|) - \ln(2) \tag{5.37}$$

de forma similar, a lo visto anteriormente si $s\delta r_{ij}$ es par, se tiene:

$$|wr_{ij}^1| = \ln(2) + |\ln(1 - |e^{-|w\delta r_{ij}|}|)| = \ln(2) + f_-(|w\delta r_{ij}|, 0) \tag{5.38}$$

y si $s\delta r_{ij}$ es impar:

$$|wr_{ij}^1| = \ln(2) - |\ln(1 + |e^{-|w\delta r_{ij}|}|)| = \ln(2) - f_+(|w\delta r_{ij}|, 0) \tag{5.39}$$

En la tabla 5.2 se presenta un resumen de las operaciones del paso horizontal.

Paso horizontal 1	$\|w\delta q_{ij}\| = \text{mín}(\|wq_{ij}^a\|) + f_-(\|wq_{ij}^0\|, \|wq_{ij}^1\|)$ $s_{ij} = 0$ si $\|wq_{ij}^0\| \leq \|wq_{ij}^1\|$ sino $s_{ij} = 1$
Paso horizontal 2	$\|w\delta r_{ij}\| = \sum_{j \in N(i)} \|w\delta q_{ij}\| - \|w\delta q_{ij}\|$ $s\delta r_{ij} = \sum_{j \in N(i)} s_{ij} - s_{ij}$
Paso horizontal 3	si $s\delta r_{ij}$ es par: $\|wr_{ij}^0\| = \ln(2) - f_+(\|w\delta r_{ij}\|, 0)$ $\|wr_{ij}^1\| = \ln(2) + f_-(\|w\delta r_{ij}\|, 0)$ si $s\delta r_{ij}$ es impar: $\|wr_{ij}^0\| = \ln(2) + f_-(\|w\delta r_{ij}\|, 0)$ $\|wr_{ij}^1\| = \ln(2) - f_+(\|w\delta r_{ij}\|, 0)$

Tabla 5.2: Resumen de las operaciones del paso horizontal

Paso vertical

Para cada i y j con $a = \{0, 1\}$ se estiman:

$$Q_{ij}^a = \alpha_{ij} \cdot f_j^a \cdot \prod_{i' \in M(j) \backslash i} R_{i'j}^a \tag{5.40}$$

donde $M(j)$ representa el conjunto de subíndices de todos los nodos paridad $h(i)$ que participan del nodo símbolo $d(j)$. Al igual que en el algoritmo de Mackay-Neal la constante α_{ij} es elegida de forma tal que se cumpla:

$$Q_{ij}^0 + Q_{ij}^1 = 1 \tag{5.41}$$

para simplificar este cálculo, en forma similar a la ecuación 2.66 se define la variable C_{ij}^a como:

$$C_{ij}^a = e^{-|wc_{ij}^a|} = f_j^a \cdot \prod_{i' \in M(j) \setminus i} R_{i'j}^a \tag{5.42}$$

si se toma a $f_j^a = e^{-|wf_j^a|}$ y $R_{i'j}^a = e^{-|wr_{i'j}^a|}$ y aplicando logaritmos se obtiene:

$$\ln(C_{ij}^a) = \ln(e^{-|wc_{ij}^a|}) = \ln(e^{-|wf_j^a|}) + \sum_{i' \in M(j) \setminus i} \ln(e^{-|wr_{i'j}^a|}) \tag{5.43}$$

quedando:

$$|wc_{ij}^a| = |wf_j^a| + \sum_{i' \in M(j) \setminus i} |wr_{i'j}^a| \tag{5.44}$$

es decir:

$$|wc_{ij}^a| = |wf_j^a| + \sum_{i \in M(j)} |wr_{ij}^a| - |wr_{ij}^a| \tag{5.45}$$

De la ecuación 2.72 del algoritmo de MacKay-Neal surge que:

$$Q_{ij}^0 = \frac{C_{ij}^0}{C_{ij}^0 + C_{ij}^1} \tag{5.46}$$

es decir:

$$Q_{ij}^0 = e^{-|wq_{ij}^0|} = \frac{e^{-|wc_{ij}^0|}}{e^{-|wc_{ij}^0|} + e^{-|wc_{ij}^1|}} \tag{5.47}$$

y aplicando logaritmos:

$$|wq_{ij}^0| = |wc_{ij}^0| + \ln(e^{-|wc_{ij}^0|} + e^{-|w_{ij}^1|}) \tag{5.48}$$

si se utilizan los resultados de las ecuaciones A.16 y A.17 y en la tabla A.1 de la sección A, se tiene que:

$$|wq_{ij}^0| = |wc_{ij}^0| - \text{mín}(|wc_{ij}^a|) + f_+(|wc_{ij}^0|, |wc_{ij}^1|) \tag{5.49}$$

donde $f_+(|wc_{ij}^0|, |wc_{ij}^1|)$ es una tabla de búsqueda con entradas $||wc_{ij}^0| - |wc_{ij}^1||$. De forma similar se obtiene $|wq_{ij}^1|$ como:

$$|wq_{ij}^1| = |wc_{ij}^1| - \text{mín}(|wc_{ij}^a|) + f_+(|wc_{ij}^0|, |wc_{ij}^1|) \tag{5.50}$$

es decir:

$$|wq_{ij}^1| = |wq_{ij}^0| + |wc_{ij}^1| - |wc_{ij}^0| \tag{5.51}$$

Luego al igual que en el algoritmo de MacKay-Neal se realiza una estimación a posteriori de las probabilidades Q_j^0 y Q_j^1 utilizando:

$$Q_j^a = \alpha_j \cdot f_j^a \prod_{i \in M(j)} R_{ij}^a \tag{5.52}$$

donde α_j se la elige de forma tal que se cumpla con:

$$Q_j^0 + Q_j^1 = 1 \tag{5.53}$$

La ecuación 5.52 se puede expresar como:

$$Q_j^a = \alpha_j \cdot f_j^a \cdot (\prod_{i' \in M(j) \backslash i} R_{i'j}^a) \cdot R_{ij}^a \tag{5.54}$$

y usando C_{ij}^a definida en la ecuación 5.42, Q_j^a queda como:

$$Q_j^a = \alpha_j \cdot C_{ij}^a \cdot R_{ij}^a \tag{5.55}$$

Usando los resultados de las ecuaciones 2.78, 2.79 y 2.80; Q_j^0 se obtiene:

$$Q_j^0 = \frac{C_{ij}^0 \cdot R_{ij}^0}{C_{ij}^0 \cdot R_{ij}^0 + C_{ij}^1 \cdot R_{ij}^1} \tag{5.56}$$

o:

$$Q_j^0 = e^{-|wq_j^0|} = \frac{e^{-|wc_{ij}^0|} \cdot e^{-|wr_{ij}^0|}}{e^{-|wc_{ij}^0|} \cdot e^{-|wr_{ij}^0|} + e^{-|wc_{ij}^1|} \cdot e^{-|wr_{ij}^1|}} \tag{5.57}$$

y aplicando logaritmo:

$$|wq_j^0| = |wc_{ij}^0| + |wr_{ij}^0| + \ln(e^{-(|wc_{ij}^0|+|wr_{ij}^0|)} + e^{-(|wc_{ij}^1|+|wr_{ij}^1|)}) \tag{5.58}$$

para agrupar términos, se define la variable $|wc_j^a|$ como:

$$|wc_j^a| = |wc_{ij}^a| + |wr_{ij}^a| \tag{5.59}$$

y usando el valor de $|wc_{ij}^a|$ definido en la ecuación 5.45, se puede expresar como:

$$|wc_j^a| = |wf_j^a| + \sum_{i \in M(j)} |wr_{ij}^a| \tag{5.60}$$

Nuevamente, utilizando los resultados de la ecuación A.17 y la tabla A.1 de la sección A, $|wq_j^0|$ resulta ser:

$$|wq_j^0| = |wc_j^0| - \text{mín}(|wc_j^a|) + f_+(|wc_j^0|, |wc_j^1|) \tag{5.61}$$

de forma similar se obtiene:

$$|wq_j^1| = |wc_j^1| - \text{mín}(|wc_j^a|) + f_+(|wc_j^0|, |wc_j^1|) \tag{5.62}$$

es decir:

$$|wq_j^1| = |wq_j^0| + |wc_j^1| - |wc_j^0| \tag{5.63}$$

Por último se puede realizar una estimación para cada valor de símbolo $d_j$usando:

$$\hat{d}_j = \text{máx}(Q_j^a) \tag{5.64}$$

es decir:

$$\hat{d}_j = 0 \quad si \quad Q_j^0 > Q_j^1 \quad sino \quad \hat{d}_j = 1 \tag{5.65}$$

y como:

$$Q_j^0 = e^{-|wq_j^0|} \quad y \quad Q_j^1 = e^{-|wq_j^1|} \tag{5.66}$$

la ecuación 5.65 puede equivalentemente expresarse como:

$$\hat{d}_j = 0 \quad si \quad |wq_j^0| < |wq_j^1| \quad sino \quad \hat{d}_j = 1 \tag{5.67}$$

también para ahorrar cálculo se puede utilizar:

$$\hat{d}_j = 0 \quad si \quad |wc_j^0| < |wc_j^1| \quad sino \quad \hat{d}_j = 1 \tag{5.68}$$

en la tabla 5.3 se muestra un resumen de las operaciones involucradas en el algoritmo de suma-resta propuesto.

Algoritmo suma-producto	Algoritmo suma-resta
Inicialización $Q_{ij}^a = f_j^a$ donde $a = \{0, 1\}$	$\|wq_{ij}^0\| = \|wf_j^0\|$, $\|wq_{ij}^1\| = \|wf_j^1\|$
Paso horizontal $R_{ij}^a = e^{-\|wr_{ij}^a\|}$	
Paso horizontal 1	$\|w\delta q_{ij}\| = \text{mín}(\|wq_{ij}^a\|) + f_-(\|wq_{ij}^0\|, \|wq_{ij}^1\|)$ $s_{ij} = 0$ si $\|wq_{ij}^0\| \leq \|wq_{ij}^1\|$ sino $s_{ij} = 1$

Tabla 5.3: Algoritmo de suma-resta (continua en la próxima página).

Algoritmo suma-producto	Algoritmo suma-resta
Paso horizontal 2	$\lvert w\delta r_{ij}\rvert = \sum_{j\in N(i)} \lvert w\delta q_{ij}\rvert - \lvert w\delta q_{ij}\rvert$ $s\delta r_{ij} = \sum_{j\in N(i)} s_{ij} - s_{ij}$
Paso horizontal 3	si $s\delta r_{ij}$ es par: $\lvert wr^0_{ij}\rvert = \ln(2) - f_+(\lvert w\delta r_{ij}\rvert, 0)$ $\lvert wr^1_{ij}\rvert = \ln(2) + f_-(\lvert w\delta r_{ij}\rvert, 0)$ si $s\delta r_{ij}$ es impar: $\lvert wr^0_{ij}\rvert = \ln(2) + f_-(\lvert w\delta r_{ij}\rvert, 0)$ $\lvert wr^1_{ij}\rvert = \ln(2) - f_+(\lvert w\delta r_{ij}\rvert, 0)$
Paso vertical 1 $Q^a_{ij} = e^{-\lvert wq^a_{ij}\rvert}$	$\lvert wc^a_{ij}\rvert = \lvert wf^a_j\rvert + \sum_{i\in M(j)} \lvert wr^a_{ij}\rvert - \lvert wr^a_{ij}\rvert$ $\lvert wq^0_{ij}\rvert = \lvert wc^0_{ij}\rvert - \text{mín}(\lvert wc^a_{ij}\rvert) + f_+(\lvert wc^0_{ij}\rvert, \lvert wc^1_{ij}\rvert)$ $\lvert wq^1_{ij}\rvert = \lvert wc^1_{ij}\rvert - \text{mín}(\lvert wc^a_{ij}\rvert) + f_+(\lvert wc^0_{ij}\rvert, \lvert wc^1_{ij}\rvert)$
Paso vertical 2 (Estimación a posteriori) $Q^a_j = e^{-\lvert wq^a_j\rvert}$	$\lvert wc^a_j\rvert = \lvert w\, f^a_j\rvert + \sum_{i\in M(j)} \lvert wr^a_{ij}\rvert$ $\lvert wq^0_j\rvert = \lvert wc^0_j\rvert - \text{mín}(\lvert wc^a_j\rvert) + f_+(\lvert wc^0_j\rvert, \lvert wc^1_j\rvert)$ $\lvert wq^1_j\rvert = \lvert wc^1_j\rvert - \text{mín}(\lvert wc^a_j\rvert) + f_+(\lvert wc^0_j\rvert, \lvert wc^1_j\rvert)$
Estimación: $\hat{d}_j = \text{máx}(Q^a_j)$	$\hat{d}_j = 0$ si $\lvert wq^0_j\rvert < \lvert wq^1_j\rvert$, sino $\hat{d}_j = 1$

Tabla 5.3: Algoritmo de suma-resta.

5.2.2. Implementación de las tablas de búsqueda

La exactitud del algoritmo de decodificación está determinada por las características que poseen las tablas de búsqueda $f_+(|a|,|b|)$ y $f_-(|a|,|b|)$.

Si el número máximo de bits usado en la construcción de la tabla es c, entonces, la tabla tendrá $N = 2^c$ entradas; y el incremento de la tabla es cada I valores, donde $I = 2^d$ es una potencia de dos. En la tabla 5.4 se muestra como queda especificada la tabla de búsqueda.

$entrada$	$salida$
0	$salida_0$
I	$salida_1$
$2I$	$salida_2$
$\cdots$	$\cdots$
N	$salida_N$

Tabla 5.4: Tabla de búsqueda.

Una forma práctica de realizar la tabla, es dividir la *entrada* por I y tomar solo la parte entera del cociente, es decir:

$$entrada^* = parte\ entera\left(\frac{entrada}{I}\right) \tag{5.69}$$

de esta forma la tabla queda ordenada en forma secuencial. Ahora, el número máximo de entradas es:

$$número\ de\ entradas^* = parte\ entera\left(\frac{N}{I}\right) \tag{5.70}$$

entonces:

$$número\ de\ entradas^* = parte\ entera\left(\frac{2^c}{2^d}\right) = 2^{c-d} \tag{5.71}$$

En la tabla 5.5 muestra como queda la nueva tabla de búsqueda . Como puede ob-

$entrada^*$	$salida$
0	$salida_0$
1	$salida_1$
2	$salida_2$
$\cdots$	$\cdots$
2^{c-d}	$salida_N$

Tabla 5.5: Tabla de búsqueda práctica.

servarse, la exactitud de la tabla queda determinada por el valor $(c-d)$ utilizado.

5.2.3. Análisis de la complejidad de la implementación del algoritmo de decodificación

En esta sección se analiza el número de operaciones involucradas en el algoritmo para tener una idea del costo que supone su implementación.

$$\mathbf{H} = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

En la matriz anterior N es el número de columnas que tiene la matriz (es decir el número de nodos símbolos-bits) y M el número de filas de la matriz (es decir el número de nodos paridad).

$N(i)$ representa el conjunto de subíndices de todos los nodos símbolos $d(j)$ que participan del nodo de paridad $h(i)$, mientras que $N(i)\backslash j$ indica la exclusión del nodo j de ese conjunto.

Por ejemplo tomando la primera fila se ve que $N(1) = 6$, es decir, en la primera fila hay 6 elementos diferentes de cero que participan del nodo paridad $h(1)$.

$M(j)$ representa el conjunto de subíndices de todos los nodos paridad $h(i)$ que participan del nodo símbolo $d(j)$. Si se toma la segunda columna se ve que $M(2) = 3$. En promedio las columnas de la matriz H tienen 3 unos, por tanto $M(j)_{promedio} = 3$ y seis unos por fila, es decir $N(i)_{promedio} = 6$.

A continuación se analiza la complejidad de la implementación del algoritmo de Mackay-Neal o suma-producto y el algoritmo de suma-resta propuesto

Algoritmo de Mackay-Neal

Si se define a $t = M(j)_{promedio}$ y $u = N(i)_{promedio}$, según [18, 27], se tiene $u = Nt/M$ y $t = 3$. Si se utiliza un código de tasa $R_c = 1/2$, es decir $M = N/2$, en este caso resulta $u = 2t = 6$. Tomando en cuenta lo dicho anteriormente, ahora se analizará la complejidad del algoritmo de Mackay-Neal.

Multiplicaciones De la ecuaciones 2.62, 2.65 y 2.74 de la subsección 2.5.6 surge:

1. Si $\delta R_{ij} = \prod_{j' \in N(i)\backslash j} \delta Q_{ij'}$, cada elemento tiene asociado $u - 2$ multiplicaciones (paso horizontal).

2. $Q_{ij}^a = \alpha_{ij}\, f_j^a \prod_{i' \in M(j)\setminus i} R_{i'j}^a$, tiene asociado $t-2$ multiplicaciones (paso vertical, estimación a priori).

3. $Q_j^a = \alpha_j\, f_j^a \prod_{i \in M(j)} R_{ij}^a$, también tiene asociado $t-2$ multiplicaciones (paso vertical, estimación a posteriori).

Para cada elemento no cero de la matriz H se tienen $(u-2)+(t-2)+(t-2)=6$ multiplicaciones asociadas. Como cada columna tiene t unos, se tienen N columnas y $a=\{0,1\}$, entonces el número total de multiplicaciones en promedio es $12Nt = 36N$.

Sumas-restas De la ecuaciones 2.62, 2.63 y 2.64 de la subsección 2.5.6 se tiene:

1. $\delta Q_{ij} = Q_{ij}^0 - Q_{ij}^1$, tiene asociada una resta.

2. $R_{ij}^0 = \frac{1}{2}(1+\delta R_{ij})$, una suma.

3. $R_{ij}^1 = \frac{1}{2}(1-\delta R_{ij})$, una resta.

4. La constante α_{ij} es elegida de forma tal que se cumpla que: $Q_{ij}^0 + Q_{ij}^1 = 1$, entonces tiene asociada una suma.

5. α_j se la elige de forma tal que se cumpla $Q_j^0 + Q_j^1 = 1$, entonces también tiene asociada una suma.

Por cada elemento no nulo de la matriz H se tienen asociadas 5 operaciones de suma-resta, entonces el número total de sumas-restas por iteración es $5Nt = 15N$.

Cocientes De la ecuaciones 2.66, 2.71, 2.74 y 2.79 de la subsección 2.5.6 surge:

1. $\alpha_{ij} = \frac{1}{C_{ij}^0 + C_{ij}^1}$ tiene asociado un cociente (paso vertical, estimación a priori).

2. $\alpha_j = \frac{1}{C_{ij}^0 \cdot R_{ij}^0 + C_{ij}^1 \cdot R_{ij}^1}$ tiene asociado un cociente (paso vertical, estimación a posteriori).

Para cada elemento no cero de la matriz H se tienen 2 cocientes asociados. Por tanto el número total de cocientes en promedio es $2Nt = 6N$

Entonces el algoritmo de MacKay-Neal presenta $36N$ multiplicaciones, $15N$ sumas-restas y $6N$ cocientes.

Algoritmo suma-resta (método propuesto)

De la ecuaciones 5.21, 5.24, 5.26 y la tabla 5.2 de la subsección 5.2.1 en el paso horizontal se tiene:

1. $|w\delta q_{ij}| = \min(|wq_{ij}^0|, |wq_{ij}^1|) + f_-(|wq_{ij}^0|, |wq_{ij}^1|)$, tiene asociada una suma, una comparación y una llamada a una tabla de búsqueda.

2. $|w\delta r_{ij}| = \sum_{j' \in N(i) \setminus j} |w\delta q_{ij'}|$, tiene $u - 2$ sumas.

3. $s\delta r_{ij} = \sum_{j' \in N(i) \setminus j} s_{ij'}$, tiene $u - 2$ sumas. (paso horizontal).

4. $|wr_{ij}^0| = \ln(2) \mp f_\pm(|w\delta r_{ij}|, 0)$, tiene una suma-resta y una llamada a una tabla de búsqueda.

5. $|wr_{ij}^1| = \ln(2) \pm f_\mp(|w\delta r_{ij}|, 0)$, tiene una suma-resta y una llamada a una tabla de búsqueda.

En el paso vertical y recurriendo a las ecuaciones 5.44, 5.49, 5.50, 5.59, 5.61 y 5.62, se tiene:

1. $|wc_{ij}^a| = |w\, f_j^a| + \sum_{i' \in M(j) \setminus i} |wr_{i'j}^a|$, tiene asociada $(t-2)+1 = t-1$ sumas, pero como $a = \{0, 1\}$ quedan $2 \cdot t - 2$ sumas.

2. $|wq_{ij}^0| = |wc_{ij}^0| - \min(|wc_{ij}^0|, |wc_{ij}^1|) + f_+(|wc_{ij}^0|, |wc_{ij}^1|)$, tiene 2 sumas-restas, una comparación y una llamada a una tabla de búsqueda.

3. $|wq_{ij}^1| = |wc_{ij}^1| - \min(|wc_{ij}^0|, |wc_{ij}^1|) + f_+(|wc_{ij}^0|, |wc_{ij}^1|)$, tiene 2 sumas-restas, una comparación y una llamada a una tabla de búsqueda.

4. $|wc_j^a| = |wc_{ij}^a| + |wr_{ij}^a|$, tiene una suma, pero como $a = \{0, 1\}$ tiene entonces asociada 2 sumas.

5. $|wq_j^0| = |wc_j^0| - \min(|wc_j^0|, |wc_j^1|) + f_+(|wc_j^0|, |wc_j^1|)$, tiene 2 sumas-restas, una comparación y una llamada a una tabla de búsqueda.

6. $|wq_j^1| = |wc_j^1| - \min(|wc_j^0|, |wc_j^1|) + f_+(|wc_j^0|, |wc_j^1|)$, tiene 2 sumas-restas, una comparación y una llamada a una tabla de búsqueda.

Para poder comparar con el algoritmo de Mackay-Neal se toma $t = 3$ quedando que el número total de sumas-restas en promedio es $25Nt = 75N$, $7Nt = 21N$ búsquedas en tablas y $5Nt = 15N$ comparaciones. A pesar de requerir más sumas-restas que

operaciones	Algoritmo Mackay-Neal	Algoritmo suma-resta
sumas-restas	$15N$	$75N$
multiplicaciones	$36N$	–
cocientes	$6N$	–
comparaciones	–	$15N$
búsquedas en tabla	–	$21N$

Tabla 5.6: Análisis de la complejidad de la implementación del algoritmo de decodificación.

el algoritmo de decodificación tradicional, la complejidad de la implementación del algoritmo propuesto se ve grandemente reducida por el hecho que no utiliza productos ni cocientes; además de poder realizar estas operaciones en punto fijo.

En la tabla 5.6, se muestran las comparaciones del grado de complejidad de los dos algoritmos.

5.2.4. Resultados obtenidos

Se ha implementado el decodificador propuesto para dos tipos de códigos LDPC, uno con una matriz de chequeo de paridad $\mathbf{H_1}$ de 30 filas por 60 columnas y otra con una matriz de chequeo de paridad $\mathbf{H_2}$ de 504 filas por 1008 columnas [113]. Las tablas de búsqueda usan $c = 16$, por tanto el número de entradas puede llegar hasta $N = 2^c = 65536$, y el máximo valor de la tabla es 65535.

Para hallar la matriz de chequeo de paridad $\mathbf{H_1}$ se generaron aleatoriamente 100 matrices $\mathbf{H}$ y de todas las obtenidas se eligió la que presentó mejor performance en la tasa de error (BER), utilizando el algoritmo tradicional [18]. La performance en la tasa de error (BER) ha sido evaluada usando el algoritmo propuesto para diferentes tamaños de tablas de búsqueda, donde cada entrada es un número entero representado en formato binario de 2 bytes.

Como puede verse en la figura 5.2, al utilizar el algoritmo propuesto, con un código LDPC que use una matriz de chequeo de paridad $\mathbf{H_1}$ pequeña, no presenta una desmejora apreciable en su performance de tasa de error (BER), si el tamaño de cada una de las dos tablas de búsqueda utilizadas es de 256 entradas de 2 bytes o más grande. El uso de tablas de tablas de 512 o más entradas no muestran diferencias apreciables con respecto al uso de la función ideal.

La performance de la tasa de error de un código LDPC más práctico que usa una matriz de chequeo de paridad $\mathbf{H_2}$ de 504 filas por 1008 columnas, se muestra en la figura 5.3. Como puede verse no hay una pérdida significativa en la performance de la tasa de error al usar el algoritmo de suma-resta, si el tamaño de ambas tablas de búsqueda es de 128 o más entradas. El uso de tablas de 256 o más entradas no

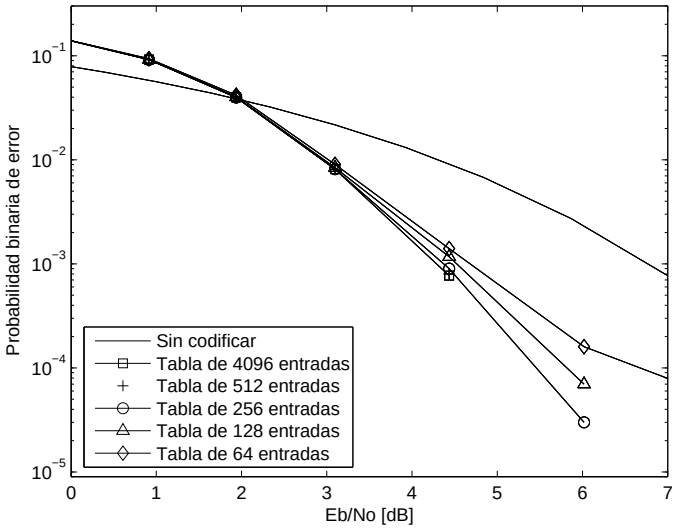

Figura 5.2: Performance del BER de un código LDPC con matriz de chequeo de paridad $\mathbf{H_1}$ de tamaño 60×30 para diferentes tamaños de tablas de búsqueda, y realizando 16 iteraciones.

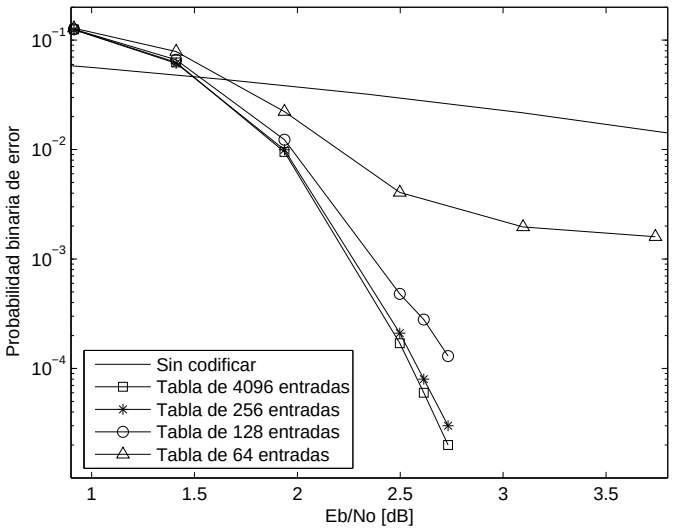

Figura 5.3: Performance del BER de un código LDPC con matriz de chequeo de paridad $\mathbf{H_2}$ de tamaño 1008×504 para diferentes tamaños de tablas de búsqueda, y realizando 16 iteraciones.

muestran diferencias apreciables con respecto al uso de la función ideal.

De esta forma es posible implementar un algoritmo de decodificación de baja complejidad sin tener una pérdida apreciable en la tasa de error, usando el algoritmo de decodificación de suma-resta propuesto con dos tablas de búsqueda de tamaño razonable.

5.2.5. Normalización con respecto al ruido

Como se puede ver en las ecuaciones 2.60 y 2.61, la probabilidad a priori f_j^a de los símbolos depende del ruido N_0 presente en el canal ($\sigma^2 = N_0/2$). Esto indica que para cada símbolo utilizado es necesario obtener el valor σ^2 del canal. En [37] se propone que es posible normalizar el algoritmo de decodificación con respecto a un valor determinado de N_0, evitando de este modo obtener información a priori del ruido del canal para cada símbolo recibido.

En base a esta idea se modificó el algoritmo del suma-resta para que trabaje con un valor fijo de ruido N_0. En la figura 5.4 se muestra el resultado obtenido al usar el algoritmo con 16 iteraciones, con una matriz $\mathbf{H}$ de 1008×504 y tablas de búsqueda de 256 valores. Como se puede apreciar, hay un determinado valor de N_0 para el cual la performance es similar al que se tiene al utilizar el N_0 propio del canal. También se aprecia que para $p_{be} = 0{,}05$ se tiene una diferencia aproximada de $0{,}4dB$ entre la performance al utilizar el N_0 del canal y $N_0 = 2$. Este valor se reduce a $0{,}3dB$ cuando $p_{be} = 10^{-3}$.

Estos resultados indican, que es posible reducir notablemente la complejidad del decodificador, debido a que no es necesario medir el N_0 del canal para cada símbolo recibido, sino solo realizar una estimación del ruido del canal periódicamente.

Los resultados obtenidos usando el decodificador de suma-resta propuesto con 16 iteraciones, son similares a los obtenidos por [34, 37], el cual tampoco mide el N_0 del canal, con una matriz $\mathbf{H}$ de 1008×504, y utilizando 50 iteraciones en su algortimo.

5.2.6. Implementación paramétrica en FPGA de un decodificador LDPC para cualquier tipo de matriz paridad y tasa de código

En esta sección se muestra la implementación de un decodificador LDPC en FPGA[114] utilizando sólo operaciones de suma-resta en punto fijo y dos tablas de búsqueda. Este decodificador puede trabajar con cualquier tamaño y tipo de matriz de paridad $\mathbf{H}$ (incluidas las generadas aleatoriamente, las generadas con una determinada regla de construcción, como las quasi-cíclicas [33, 115], sean de tipo sistemático,

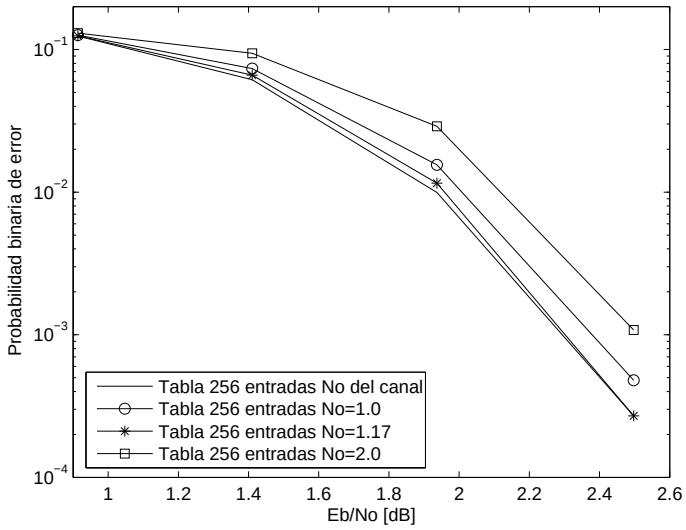

Figura 5.4: Performance de un decodificador LDPC de 16 iteraciones, que usa tablas de 256 valores, $\mathbf{H}$ de 1008×504 para diferentes valores de ruido del canal N_0.

o no sistemático) y su arquitectura puede ser fácilmente configurada para diferentes tasas de código. Los resultados obtenidos muestran que es posible implementar decodificadores en lógica programable de muy baja complejidad con muy buena performance [111].

Arquitectura del decodificador

En la figura 5.5 se observa el decodificador implementado. La memoria *ROM_H* guarda la ubicación de cada uno en la matriz $\mathbf{H}$; su tamaño depende del número de columnas con unos en cada fila y del número de filas que posee. Por ejemplo, para una matriz $\mathbf{H}_1$ generada aleatoriamente, de 60×30, el tamaño de *ROM_H* es de 256 palabras de 6 bits, y para una matriz $\mathbf{H}_2$ generada aleatoriamente, de 1008×504, es de 4096 palabras de 10 bits.

La *ROM_num_unos_fil* almacena el número de columnas con unos que tiene cada fila. Esta memoria se utiliza para poder indexar y poder así recuperar junto con la *ROM_H*, la posición de cada uno en la matriz $\mathbf{H}$.

Las memorias *ROM_ftabla+* y *ROM_ftabla-* contienen las tablas de búsqueda $f_+(a)$ y $f_-(a)$. Ambas tablas son de 256 palabras de 16 bits, tanto para $\mathbf{H}_1$ como para $\mathbf{H}_2$.

Las RAM_wf_0 y RAM_wf_1 almacenan a $|wf_0|$ y $|wf_1|$ respectivamente cada vez que ingresa una nueva palabra al decodificador. Su tamaño depende de la longitud de palabra utilizada.

Las memorias RAM_wq_0 y RAM_wq_1, poseen igual cantidad de palabras que la

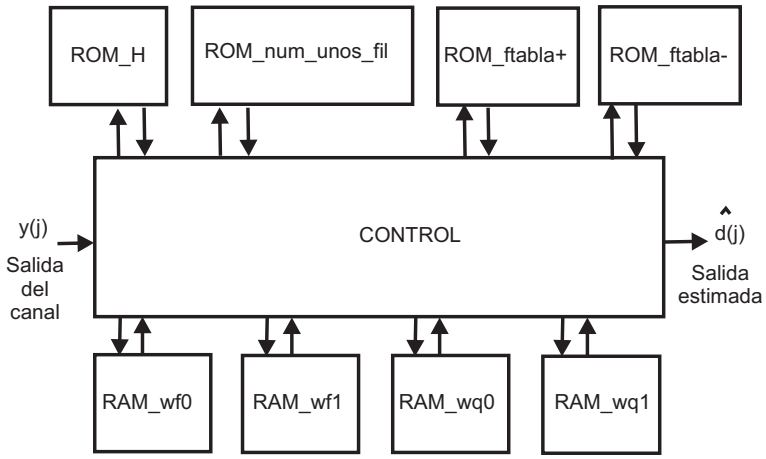

Figura 5.5: Arquitectura del decodificador LDPC implementado en FPGA de ALTERA [1]

memoria *ROM_H*, pero son de 16 bits. Las RAM_wq_0 y RAM_wq_1 realizan varias funciones: En la inicialización almacenan los valores de $\left|wq_{ij}^0\right|$ y $\left|wq_{ij}^1\right|$ respectivamente. En el paso horizontal 1, RAM_wq_0 guarda los valores de $|w\delta q_{ij}|$ y RAM_wq_1 guarda los valores de s_{ij}. En el paso horizontal 2 se almacenan los valores de $|w\delta r_{ij}|$ y $s\delta r_{ij}$. En el paso horizontal 3 se almacenan los valores de $\left|wr_{ij}^0\right|$ y $\left|wr_{ij}^1\right|$. Finalmente en el paso vertical se vuelven a almacenar los valores de $\left|wq_{ij}^0\right|$ y $\left|wq_{ij}^1\right|$. Esto le da un alto grado de reutilización a estas memorias.

En la tabla 5.7 se indica el tamaño de memorias usadas en cada decodificador.

Resultados obtenidos

Ambos decodificadores LDPC (60×30) y LDPC (1008×504) se implementaron usando lenguaje VHDL [74, 75] y la herramientas de síntesis del programa QUARTUS II [116] de ALTERA. En la tabla 5.8 se pueden ver los resultados obtenidos en la implementación de ambos decodificadores.

Memoria	LDPC (60×30)	LDPC (1008×504)
RAM_wf	64×16	1024×16
ROM_H	256×6	4096×10
RAM_wq	256×16	4096×16
ROM_tabla	256×16	256×16
ROM_num_unos_fil	32×3	512×4

Tabla 5.7: Memorias utilizadas en la implementación

Hardware utilizado	LDPC (60×30)	LDPC (1008×504)
Dispositivo	EP2C5T144C6	EP2C20F256C6
Familia	Cyclone II	Cyclone II
Elementos lógicos	976	1047
Registros	388	425
Bits de memoria	15968	210944
Frecuencia del reloj	$71,81Mhz$	$69,49Mhz$

Tabla 5.8: Resultados obtenidos en la implementación

5.3. Implementación de un decodificador LDPC usando programación lineal PL

A continuación se propone la realización de un decodificador LDPC que utiliza programación lineal. Primeramente se analiza su performance, comparándola con el algoritmo de suma-producto clásico y luego como es posible lograr su implementación en lógica programable.

Para resolver el problema de programación lineal, se utiliza la función *LinProg* del programa Matlab®. Esta función resuelve el siguiente sistema de PL:

$$\begin{aligned} min_x \quad & \mathbf{c^T} \cdot \mathbf{x} \\ s \cdot a \cdot \quad & \mathbf{Ax} \leq \mathbf{b} \\ & l_{inf} \leq \mathbf{x} \leq l_{sup} \end{aligned} \tag{5.72}$$

Se utilizó un decodificador con una matriz de chequeo de paridad **H** de 30 filas por 60 columnas, la cual permite trabajar con mensajes de 60 bits. Todos las simulaciones fueron realizadas utilizando un canal con ruido blanco Gaussiano.

Se realizó una comparación entre el algoritmo tradicional de suma-producto y el decodificador basado en el uso de programación lineal. En la decodificación por PL se usaron dos niveles de umbral.

En el primer umbral, se reconoce que $y_i = 1$ si $f_i \geq 0{,}5$ e $y_i = 0$ si $f_i < 0{,}5$; en el segundo umbral, $y_i = 1$ si $f_i \geq 0{,}99$ e $y_i = 0$ si $f_i < 0{,}01$.

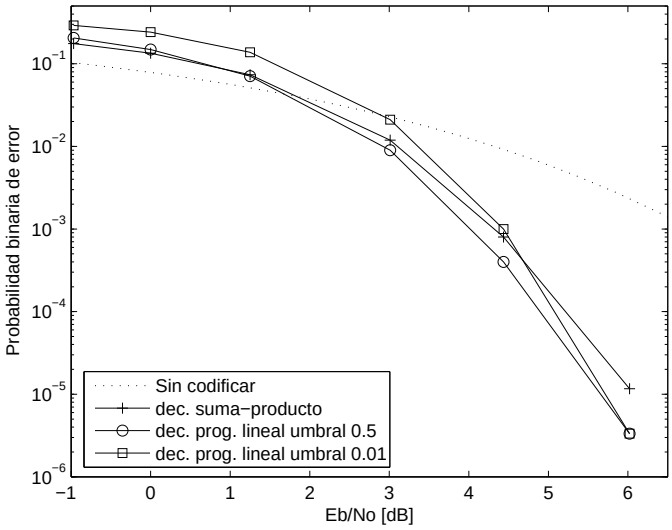

Figura 5.6: BER para un código LDPC (30,60).

Como se puede apreciar en la Figura 5.6, no se observan diferencias notables entre el algoritmo tradicional y el de programación lineal. No obstante, se observa un pequeño aumento del error al utilizar un umbral más ajustado.

5.3.1. Implementación práctica

Tratar de implementar un decodificador en lógica programable que utilice programación lineal no es tarea sencilla. No obstante, en [117], se presenta un algoritmo que permite decodificar en forma práctica utilizando PL.

El núcleo del decodificador se basa en operaciones que se realizan utilizando el siguiente operador:

$$\min_{l}^{(\lambda)} z_l \triangleq -\frac{1}{\lambda}\, ln\left(\sum_{l} e^{-\lambda z_l}\right) \tag{5.73}$$

De acuerdo a la ecuación A.9 se puede poner:

$$\ln\left(e^{wa} + e^{wb}\right) = \text{máx}(wa, wb) + \ln(1 + e^{|wa-wb|}) \tag{5.74}$$

El término $\ln(1 + e^{|wa-wb|})$ es un factor de corrección. Éste puede estar contenido en una tabla de búsqueda $f_+(|wa|, |wb|)$, con entradas $|wa - wb|$. Entonces:

$$\ln\left(e^{wa} + e^{wb}\right) = \text{máx}(wa, wb) + f_+(|wa|, |wb|) = g(wa, wb) \tag{5.75}$$

En base al resultado obtenido en la ecuación 5.75, la ecuación 5.73 queda:

$$-\frac{1}{\lambda}\, ln\left(\sum_{l} e^{-\lambda z_l}\right) = -\frac{1}{\lambda}\, g(z_k.g(z_{\lambda-1} \ldots g(z_3.g(z_2, z_1))) \ldots) \tag{5.76}$$

es decir:

$$\min_{l}^{(\lambda)} z_l = -\frac{1}{\lambda}\, g(z_\lambda.g(z_{\lambda-1} \ldots g(z_3.g(z_2, z_1))) \ldots) \tag{5.77}$$

Con este resultado, el algoritmo propuesto por [117] puede ser fácilmente implementado en lógica programable, debido a que para efectuar la operación $\text{mín}_l^{(\lambda)}\, z_l$ sólo es necesario el uso de un comparador y una tabla de búsqueda.

5.4. Conclusiones

Se propuso una modificación al algoritmo de decodificación de códigos de paridad de baja densidad conocido como el algoritmo de MacKay-Neal [18] (sección 2.5.6). El algoritmo propuesto permite utilizar tablas de búsqueda de un tamaño aceptable y operaciones de suma y restas para construir decodificadores en lógica programable de muy baja complejidad, ya que no es necesario utilizar aritmética de punto flotante, ni realizar productos ni cocientes.

Utilizando este algoritmo se ha implementado en FPGA un decodificador LDPC, el cual puede trabajar con cualquier tamaño y tipo de matriz de paridad **H** (incluidas las generadas aleatoriamente, las generadas con una determinada regla de construcción, como las quasi-cíclicas, sean de tipo sistemático, o no sistemático), y se adapta en forma paramétrica cualquiera sea la tasa del código k/n.

El sistema obtenido muestra una gran versatilidad de aplicación y no presenta diferencias apreciables en la performance de tasa de error con respecto al uso del decodificador de suma-producto ideal introducido por D. J. C. MacKay y R. M. Neal [18, 26].

Luego se mostró que un decodificador LDPC que emplee programación lineal presenta una performance similar a la obtenida con un decodificador suma-producto clásico.

Si bien su implementación utilizando el algoritmo de programación lineal clásico no es sencilla, es posible realizar una implementación práctica, en la cual operaciones complejas son reemplazadas por comparaciones y búsqueda en tablas.

Capítulo 6

Implementaciones de sistemas de control de error para enlaces de alta complejidad con alto grado de privacidad

6.1. Introducción

Hoy en día, en las comunicaciones inalámbricas y móviles, hay una gran necesidad de diseñar sistemas de comunicaciones con un alto grado de privacidad y una excelente performance en la tasa de error (BER). La transmisión de información encriptada en un canal afectado por ruido, produce una propagación de errores, que degrada la performance de la tasa de error (BER) total del sistema.

En este capítulo, se presentan esquemas que combinan el uso del algoritmo de encriptación AES con códigos muy eficientes para el control de errores, como los códigos LDPC y turbo [118, 119, 120].

En estos esquemas, el nivel de privacidad se puede mejorar notablemente usando procedimientos de permutación de datos de naturaleza pseudo-aleatoria en los codificadores y decodificadores. Estas permutaciones se pueden implementar prácticamente usando llaves de *hardware*.

Así, combinando eficientes técnicas de control de errores con el algoritmo AES, es posible obtener un sistema de comunicación con una gran capacidad de encriptamiento y una excelente performance en la tasa de error (BER).

Por último se ilustra cómo es posible implementar fácilmente un sistema en lógica programable usando un decodificador LDPC que utiliza sólo sumas y restas realizadas en punto fijo y un encriptador-desencriptador AES de bloques de 128 bits y una llave de 128 bits.

6.2. Implementación de un esquema con control de error y encriptamiento combinado

Generalmente la mayoría de las técnicas de encriptamiento son diseñadas suponiendo el uso de un canal libre de ruido. Sin embargo, en el caso de los sistemas de comunicaciones inalámbricos, el canal está afectado fuertemente por el ruido [121]. Con respecto a la seguridad de la información transmitida, son bien conocidos los ataques reportados sobre las técnicas de encriptamiento implementadas en el estándar 802.11 para LAN inalámbricas, llamado protocolo WEP (*Wired Equivalent Privacy*) [122]. Por esta razón, es necesario la implementación de una mejor técnica de encriptación [47] y también tomar en cuenta el problema de la tasa de error (BER) en el canal inalámbrico [39].

Una la las técnicas de encriptación más robusta es la utilizada en el algoritmo denominado AES (*Advanced Encryption Standard*) [47]. Este algoritmo genera un conjunto de operaciones no lineales sobre el mensaje original (típicamente un bloque de 128 bits de información) que le otorgan una fuerte capacidad de encriptamiento. Sin

embargo, este mismo efecto no lineal ocasiona una fuerte propagación de errores, de manera que cuando se lo utiliza en una transmisión inalámbrica por ejemplo, y frente a la acción del ruido, la performance de tasa de error (BER) se deteriora fuertemente, si lo comparamos con la transmisión sin codificar.

En la Sección 6.2.1 se analiza el comportamiento de este algoritmo de encriptación en presencia del ruido. Se muestra que la pérdida en la performance de tasa de error (BER) es entre 1 a 5 dB con respecto a una transmisión sin codificar. Estos valores de pérdida dependen del valor de la relación de energía promedio del bit con respecto a la densidad espectral de potencia de ruido E_b/N_0 a la cual es medido. Como consecuencia de este análisis, se encuentra que es conveniente combinar este algoritmo de encriptamiento con alguna técnica de control de errores tales como los códigos LDPC o turbo. El resultado final es la obtención de un sistema con muy buena capacidad de privacidad y excelente performance de la tasa de error (BER).

El uso combinado del algoritmo AES junto con técnicas de control de errores puede ayudar a eliminar parcialmente este efecto de degradación. Para este fin se han seleccionado los códigos LDPC [18, 19] y turbo [38, 42]. Se obtiene así un sistema de comunicaciones con una excelente performance en el BER y en el nivel de privacidad.

6.2.1. Comparación entre la transmisión sin codificar y la transmisión utilizando el algoritmo AES, en un canal con ruido blanco Gaussiano

AES-128 [47] es un cifrador en bloques simétrico iterativo de llave privada que opera con un bloque de tamaño $L = 128$ bits. Las operaciones realizadas por el algortimo AES provocan transformaciones no lineales del texto plano, (en inglés *plaintext*). Esta fuerte alinealidad produce que al transmitirse por un canal con presencia de ruido, el efecto de un simple error en el paquete encriptado se propague fuertemente, de manera que un único error produce una cantidad de bits equivocados que es aproximadamente igual a la mitad de la longitud de bloque de encriptamiento.

La figura 6.1 muestra la performance del BER de una transmisión de información encriptada con AES, en comparación con una transmisión de información sin codificar, sobre un canal con Ruido Gaussiano Blanco Aditivo (RGBA), (en inglés *Additive White Gaussian Noise*, AWGN). La transmisión se realiza en ambos casos en formato binario polar. Como se ve en la figura 6.1, el uso del algoritmo AES produce una pérdida en la performance del BER con respecto a una transmisión sin codificar cuando la transmisión se realiza en un canal que presenta ruido. Esto sugiere que se debería agregar alguna técnica de codificación con control de errores para compensar esta pérdida. La pérdida en la performance del BER es entre 1 dB y 5 dB, dependiendo esto del valor de E_b/N_0 a la cual es calculada.

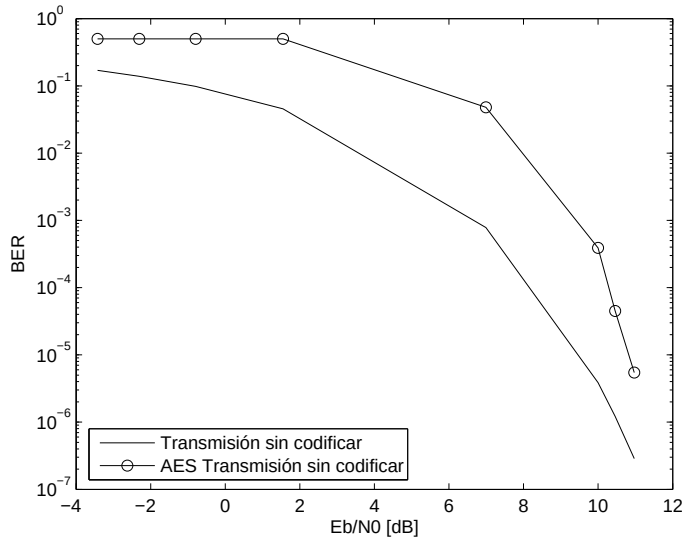

Figura 6.1: Performance del BER de una transmisión de información binaria encriptada con AES y de una transmisión de información binaria sin codificar.

Cuando la información encriptada usando AES se transmite en formato binario en un canal que presenta RGBA y ocurre un error en uno ó más bits de un elemento del campo de Galois $GF(256)$, cuando se desencripta este bloque de 128 bits erróneo, se produce una propagación de errores. Esto sucede aunque se produzca un solo bit erróneo en el bloque de 128 bits correspondiente a un elemento dado del campo de Galois $GF(256)$. Entonces al transmitir en un canal con ruido, el algoritmo AES produce una propagación de error que multiplica el número de bits errados con respecto a una transmisión sin codificar por un factor que fluctúa entre 0 y L dependiendo del tipo de evento que se produzca en el canal. En promedio, este factor es aproximadamente igual a:

$$T_{AES} \cong \frac{L}{2} = 64. \tag{6.1}$$

Para poder ver el número promedio de errores que se producen al desencriptar un bloque, se ha simulado un transmisión encriptada en AES en bloques de 128 bits en un canal en el cual se halla presente ruido [16]. En cada transmisión solo se altera un bit del bloque de 128 bits. La figura 6.2 muestra el número de errores que se producen en la información desencriptada al variar un bit en el bloque transmitido, en función del la posición del bit alterado. Los resultados de esta simulación dependen de la llave utilizada en el algoritmo AES y del mensaje transmitido. En el caso que ilustra la figura 6.2 se utilizó como llave del algoritmo de todos ceros y como mensaje se transmitió un vector de 128 ceros. El número promedio de errores producidos es igual a $T_{AES} = 64{,}1719 \cong L/2$. Esto indica que la tasa de error binaria de la transmisión

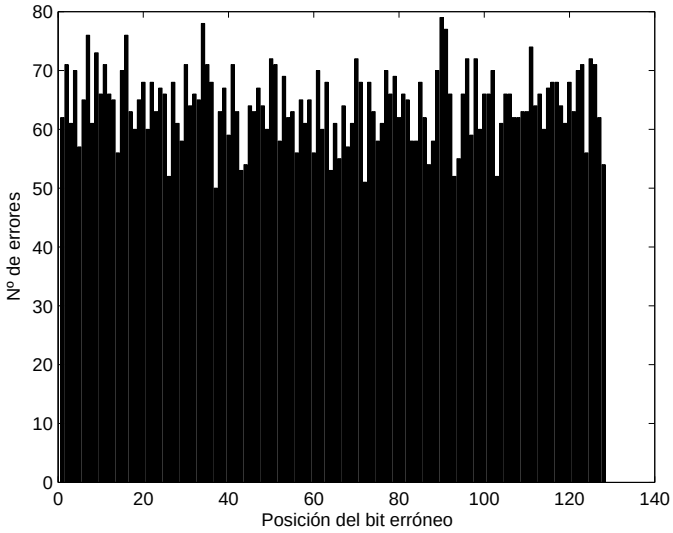

Figura 6.2: Número de errores generados al desencriptar mediante el algortimo AES bloques de 128 bits que contienen un bit erróneo cada uno, en función de la posición del bit alterado.

de información binaria encriptada usando AES p_{be_AES} es aproximadamente igual a la tasa de error binaria de la transmisión sin codificar p_{be} multiplicada por el factor de propagación de error T_{AES}. Esto puede ser verificado en la figura 6.1, donde se ve que el factor de degradación del BER es de alrededor de T_{AES}. También se observa que para bajo valores de E_b/N_0 el efecto de propagacion de error hace que la performance del BER de la transmisión de información binaria encriptada usando AES alcance su peor valor de probabilidad de error, es decir, $p_{be\_AES} \approx 0{,}5$.

6.2.2. Esquema combinado de un código LDPC con matriz de chequeo de paridad H de tamaño 256×128 y el algoritmo AES

El algoritmo AES-128, es un cifrador que opera con bloques de $L = 128$ bits y tiene una llave de la misma longitud. Por otro lado, los códigos LDPC son códigos lineales iterativos muy poderosos. Cuando los códigos LDPC operan con bloques de gran tamaño, su performance se aproxima al límite de Shannon [18, 19].

Para combinar ambas técnicas de transmisión, la salida del cifrador AES es la entrada a un codificador LDPC $C_{LDPC}\ (256, 128)$ que toma el bloque de 128 bits ya encriptado, y genera un bloque codificado con control de errores con un tamaño de 256 bits. El esquema se muestra en la figura 6.3.

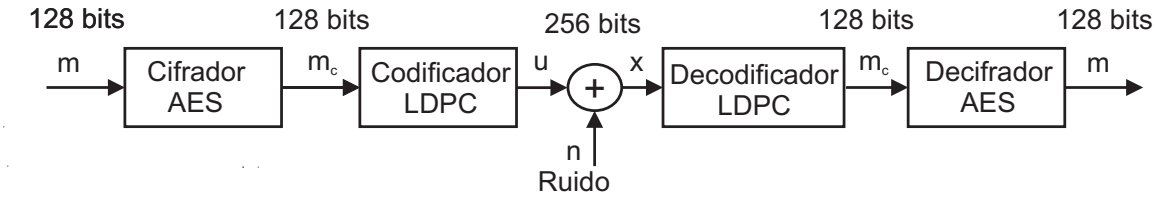

Figura 6.3: Sistema combinado, que utiliza el algortimo AES y un código LDPC $C_{LDPC}(256, 128)$.

En el esquema propuesto, primeramente la información binaria es aplicada al cifrador AES y la salida de éste ya encriptada, se aplica al codificador LDPC C_{LDPC} $(256, 128)$.

Si se aplicara en forma inversa, es decir, en el transmisor primero se aplicara la codificación para el control de errores mediante el codificador LDPC, y luego el cifrador AES; en el receptor el descifrador AES operaría primero, entregando como salida un bloque de 128 bits en formato de decisión rígida, (es decir una secuencia de'1's y '0's), produciendo la perdida de información suave obtenida del canal [18, 19].

De esta manera, se pierde la eficacia de la decodificación iterativa en este tipo de códigos, que esencialmente tienen una buena performance si operan sobre la base de entradas y salidas que sean estimaciones no rígidas. Además, el algoritmo AES opera sobre la base de información binaria y el decodificador LDPC le presenta la información en este formato, luego de haber decodificado iterativamente.

Otra razón que justifica este orden, es que en el lado receptor, el decodificador LDPC reduce los errores producidos en el canal, aliviando así el efecto de propagación de error que el desencriptador del algoritmo AES produce. De esta manera el decodificador LDPC genera la entrada al desencriptador en formato binario explotando su capacidad de control de error y la información no rígida obtenida del canal [19].

En el caso de la figura 6.3, el decodificador LDPC opera en forma usual, generando una estimación del vector decodificado luego de un determinado número de iteraciones. Este vector se transfiere al descifrador AES como un bloque de 128 bits en el formato binario clásico de unos y ceros, la cual es la entrada apropiada para este descifrador. El decodificador LDPC reduce drásticamente el número de errores en el bloque de 128 bits que ingresan al descifrador AES para ser desencriptados, con respecto al caso en que no se aplica ninguna corrección de errores.

El código LDPC utilizado en el esquema propuesto es un código $C_{LDPC}(256, 128)$, el cual tiene una tasa de código $R_c = 1/2$, y se adapta al formato de bloques que entrega el encriptador de AES. De esta forma, el codificador LDPC genera un bloque de 256 bits que contiene los bits de información encriptados y los bits de redundancia asociados. En este primer esquema, el código LDPC utiliza una matriz de chequeo

de paridad H de tamaño 128×256, la cual es de la forma $\mathbf{H} = [\mathbf{A}\ \mathbf{B}]$, donde las submatrices $\mathbf{A}$ y $\mathbf{B}$ son matrices cuadradas de tamaño 128×128.

La salida del codificador LDPC es un vector $\mathbf{u} = [\mathbf{c}\ \mathbf{m_c}]$, donde $\mathbf{m_c}$ es el mensaje cifrado a ser codificado y $\mathbf{c}$ es el vector que contiene los bits de redundancia. Por otro lado, $\mathbf{m}$ es el vector de mensaje a ser transmitido. Los bits de redundancia se calculan como:

$$\mathbf{c} = \left(\mathbf{A^{-1}} \circ \mathbf{B}\right) \circ \mathbf{m_c}^{\mathbf{T}} \tag{6.2}$$

por lo tanto, la submatriz $\mathbf{A}$ debe tener inversa.

6.2.3. Incremento en la capacidad de encriptado

En la primera versión del sistema propuesto, la matriz de chequeo de paridad $\mathbf{H}$ se mantuvo fija. Sin embargo, si se realizan permutaciones pseudo aleatorias de las columnas de la matriz $\mathbf{H}$ se puede incrementar notablemente el nivel de encriptamiento. No obstante puede suceder que luego de realizar las permutaciones de las columnas, la nueva matriz de chequeo de paridad $\mathbf{H}$, tenga una submatriz $\mathbf{A}$ que no sea inversible.

Cuando la submatriz $\mathbf{A}$ tiene inversa, la permutación de sus columnas es equivalente a realizar la misma permutación sobre los bits correspondientes del vector del código. Es decir:

$$Perm(\mathbf{c}) = \left(\left(Perm(\mathbf{A})\right)^{-1} \circ \mathbf{B}\right) \circ \mathbf{m_c}^{\mathbf{T}} \tag{6.3}$$

donde $Perm()$ indica la operación de permutar la posiciones de las columnas de la correspondiente matriz, o los bits del vector correspondiente. La regla de permutación es la misma en ambos casos. Por otro lado, si la submatriz $\mathbf{A}$ original tiene inversa, la matriz permutada $Perm(\mathbf{A})$ también resulta inversible.

Dado que realizar permutaciones sobre las columnas de la matriz de paridad $\mathbf{H}$ puede generar una matriz que posea una submatriz $\mathbf{A}$ no invertible, una solución sería aplicar las reglas de permutación en forma independiente sobre las submatrices $\mathbf{A}$ y $\mathbf{B}$. Sin embargo, esto no es similar a realizar la operación equivalente sobre las posiciones de los bits del vector del código. Si se realizan las operaciones de permutación en forma independiente sobre las submatrices $\mathbf{A}$ y $\mathbf{B}$ se tiene:

$$Perm(\mathbf{H}) = [Perm_1(\mathbf{A})\ Perm_2(\mathbf{B})] \tag{6.4}$$

pero:

$$Perm(\mathbf{c}) \neq \left(\left(Perm_1(\mathbf{A})\right)^{-1} \circ Perm_2(\mathbf{B})\right) \circ \mathbf{m_c}^{\mathbf{T}} \tag{6.5}$$

Resultaría conveniente que la operación de permutación sobre la matriz de chequeo de paridad $\mathbf{H}$ fuera equivalente a la permutación de las posiciones de los bits del vector. De otra forma luego de realizar la permutación de la matriz de chequeo de paridad $\mathbf{H}$

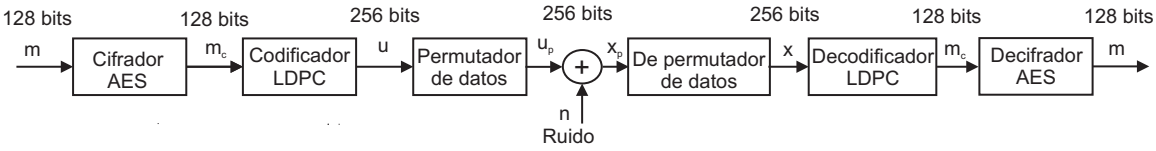

Figura 6.4: Sistema combinado, que utiliza el algortimo AES y un código LDPC $C_{LDPC}(256, 128)$ con permutador de datos.

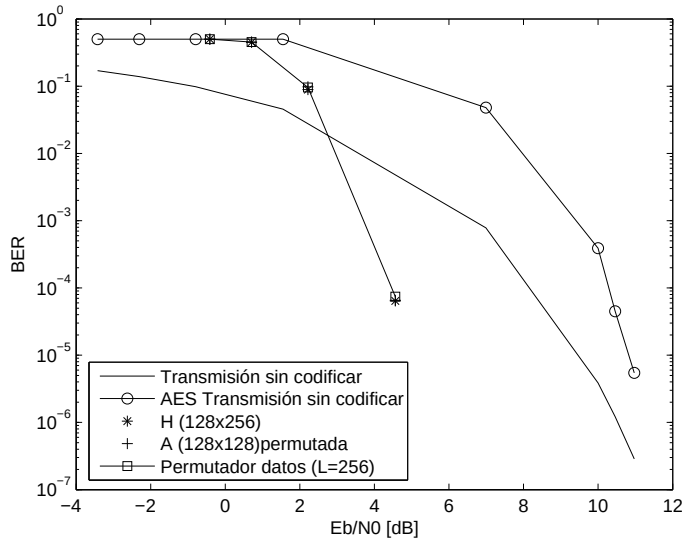

Figura 6.5: Performance del BER de un sistema combinado, que utiliza el algortimo AES y un código LDPC $C_{LDPC}(256, 128)$ con permutación pseudo aleatoria de los bits del vector de código y con permutación pseudo aleatoria de las columnas de la submatiz $\mathbf{A}$.

sería necesario calcular durante la codificación la inversa de la submatriz $(Perm(\mathbf{A}))$, $(Perm(\mathbf{A}))^{-1}$, para hallar los bits de redundancia usando la ecuación 6.2. El cálculo de la inversa de la submatriz $(Perm(\mathbf{A}))^{-1}$ es una operación muy costosa, debido al número de operaciones necesarias para realizarlo.

Por tanto, es más conveniente aplicar las permutaciones sobre las posiciones de los bits del vector de código, en vez de hacer la operación equivalente sobre las columnas de la matriz de chequeo de paridad $\mathbf{H}$. La permutación pseudo aleatoria sobre las posiciones del los bits del vector de código es mucho más simple, y es similar a la operación de permutación realizada sobre las columnas de la matriz de chequeo de paridad $\mathbf{H}$. En el receptor se realiza la operación inversa de permutación sobre los bits de la señal recibida, de forma de reordenar el vector de código original. El esquema empleado se muestra en la figura 6.4.

La figura 6.5 muestra la performance del BER de un esquema que usa una matriz

de chequeo de paridad **H** fija, la performance de otro esquema donde la submatriz **A** es permutada usando una regla pseudo aleatoria, y la performance de un esquema donde los bits del vector de código son permutados usando también una regla pseudo aleatoria. Como puede apreciarse en esa figura, las tres performances del BER son prácticamente iguales. Es decir esta operación no mejora ni degrada la performance del esquema original (el que utiliza una matriz de chequeo de paridad **H** fija). Sin embargo, esto produce un incremento en la capacidad de encriptamiento del sistema en un factor de $(2L)!$ [123].

Así, se puede obtener un incremento en la capacidad de encriptamiento mediante las permutaciones pseudo aleatorias de las posiciones de los bits del vector de código, sin degradar la correspondiente performance del BER. Dado que la longitud de los bloques utilizados es usualmente bastante grande, el incremento de la capacidad de encriptamiento es muy elevada. En este caso, el factor de incremento de la capacidad de encriptamiento resulta ser de $2L! = 256! \cong 8{,}58 \times 10^{506}$.

6.2.4. Esquema combinado de un código LDPC con matriz de chequeo de paridad H de tamaño 2560×1280 y el algoritmo AES

Como es bien conocido, los códigos LDPC son más eficientes cuanto más larga es la palabra del código. Por esta razón se propone otro esquema en el que se utiliza a la salida del cifrador AES un registro que toma 10 bloques de 128 bits cada uno, y genera una palabra de 1280 bits encriptados. Esta palabra de 1280 bits es aplicada a la entrada de un codificador $C_{LDPC}(2560, 1280)$ de tasa $R_c = 1/2$, que produce un bloque encriptado y codificado de 2560 bits. Estos bits son transmitidos en el canal para ser primero decodificados por el decodificador LDPC iterativo, y luego desencriptados por los correspondiente descifrador.

La performance de tasa de errores (BER) de este esquema se puede ver en la figura 6.7. Se observa, tal como podía esperarse, que la tasa de error de este esquema que agrupa 10 bloques de mensaje para codificar, es mejor que la del sistema propuesto en la sección 6.2.2, que codifica bloques de sólo 128 bits. La mejora en la performance para una tasa de error de 10^{-4} es de aproximadamente $2dB$. El sistema propuesto se puede observar en la figura 6.6 y ha sido implementado con y sin permutador y de-permutador de datos.

Como se ve en la figura 6.7, el uso de un permutador de datos no modifica la performance de la tasa de error (BER) con respecto a un sistema similar que no utiliza esta permutación. En este caso, el uso del permutador de datos incrementa la capacidad de encriptamiento del esquema por un factor de $20L!$. Este número resulta ser igual a $20L! = 2560! \cong 2{,}53 \times 10^{7615}$. También en la figura 6.7 se puede ver que el

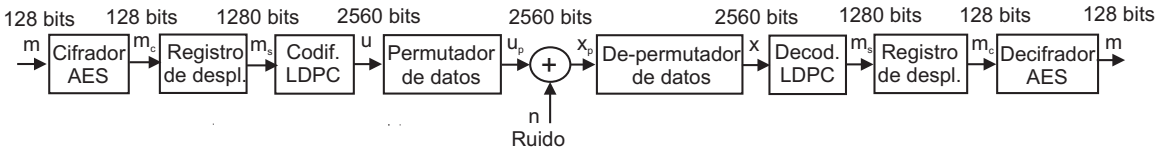

Figura 6.6: Sistema combinado, que utiliza el algortimo AES y un código LDPC $C_{LDPC}(2560, 1280)$ con permutador de datos.

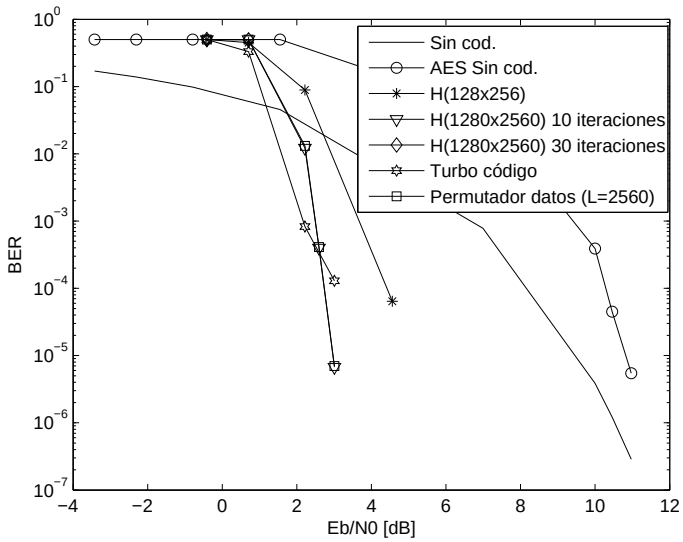

Figura 6.7: Performance de la tasa de error, de diferentes esquemas que usan encriptación AES, diferentes códigos LDPC y también código turbo.

código LDPC que utiliza 30 iteraciones tiene una tasa de error menor que 1×10^{-7} para un valor de $E_b/N_0 = 3\ dB$.

6.2.5. Esquema combinado que usa un código turbo con mezclador de datos aleatorio de longitud $L_T = 1280$ y el algoritmo AES

En este esquema se combina el algoritmo AES con un código turbo. Un grupo de 10 bloques de 128 bits de información encriptada en AES, forma un bloque de 1280 bits que es la entrada a un codificador turbo, cuyo permutador de datos aleatorio es también de longitud $L_T = 1280$. El diagrama del sistema se ilustra en la figura 6.8.

El sistema de código turbo utilizado en la figura figura 6.8 se observa en detalle en la figura 6.9. Los códigos constituyentes del codificador turbo son códigos convolucionales recursivos del tipo $(7, 5)$, $(111, 101)$, y un permutador de datos aleatorio de

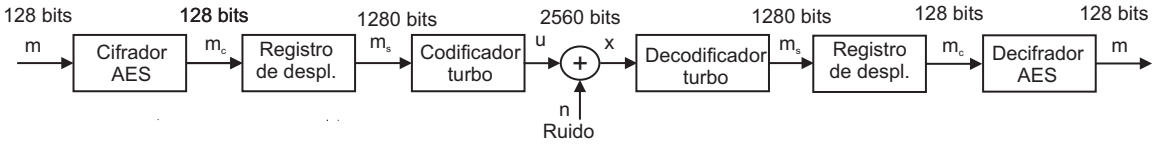

Figura 6.8: Esquema combinado que usa un código turbo con mezclador de datos aleatorio de longitud $L = 1280$ y el algoritmo AES.

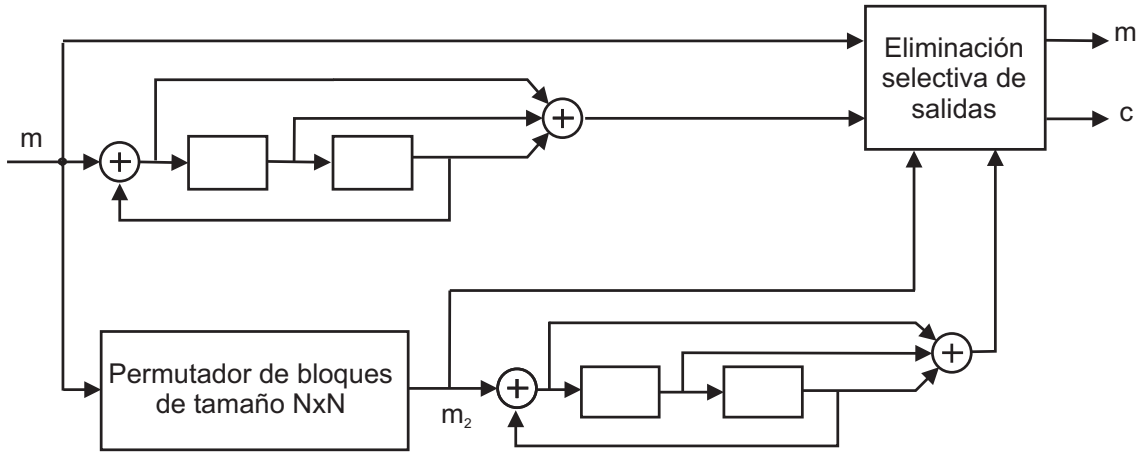

Figura 6.9: Detalle del sistema de Código Turbo utilizado.

longitud $L_T = 1280$ [9, 38, 40].

Se utiliza un mecanismo de eliminación selectiva de salidas (conocido en inglés como *puncturing*) de manera que la tasa del código turbo es $R_c = 1/2$. Los bits de mensaje no son quitados por el proceso de selección. El proceso de selección de salidas se realiza transmitiendo alternativamente una u otra de las salidas de los codificadores constituyentes, en forma alternada con los bits de mensaje que son transmitidos en forma sistemática [123]. La performance de tasa de error de este esquema puede verse en la figura 6.7. En la misma se observa que si bien es levemente mejor que la del esquema que utiliza el código LDPC $C_{LDPC}(2560, 1280)$ en regiones de valor bajo de E_b/N_0, presenta también efecto de piso para valores mas altos, donde el esquema que utiliza el código LDPC $C_{LDPC}(2560, 1280)$ funciona mejor. En todos los casos, la utilización de un codificador para el control de errores eficiente produce una mejora sustancial en la performance del sistema, si se compara esta situación con la del uso del algoritmo AES sin codificación para el control de errores. Así,mientras la perdida de la transmisión encriptada con AES es de alrededor de $1\,dB$ a $5\,dB$ con respecto de la transmisión sin codificar, el empleo de codificación para control de errores lleva al sistema a ganancias del orden de $8\,dB$, con respecto de la transmisión encriptada.

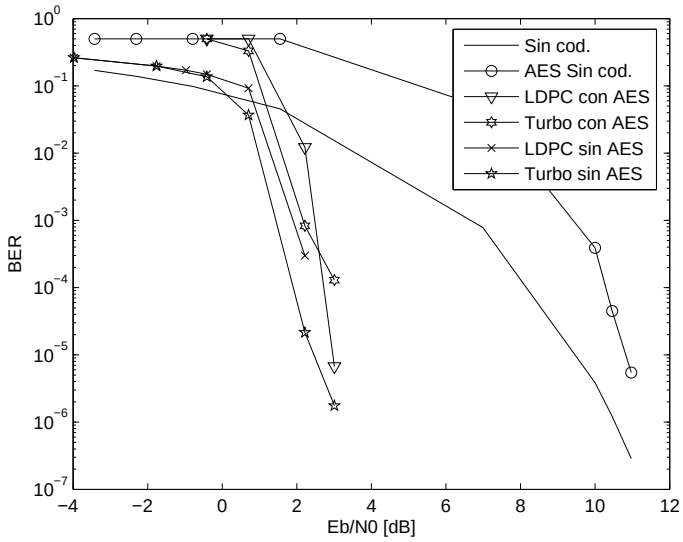

Figura 6.10: Comparación de los esquemas de encriptamiento y control de errores propuestos con esquemas similares sin la aplicación del algoritmo AES.

6.2.6. Comparación de los esquemas de encriptamiento y control de errores propuestos con esquemas similares sin aplicación del algoritmo AES

Como se vio en la sección 6.2.1, la transmisión de información binaria encriptada utilizando el algoritmo AES en un canal que presenta ruido, produce un efecto de propagación de errores. Este efecto puede ser mitigado combinando junto con el algoritmo AES el uso de codificación para el control de errores.

Se ha medido la variación en la performance de la tasa de error de los esquemas propuestos [16] operando con y sin el uso del algoritmo AES. Los resultados obtenidos se ilustran en la figura 6.10. Como se observa en esta figura, el efecto de propagación de errores que el algoritmo AES produce sobre la información, que como se dijo anteriormente puede medirse por un factor T_{AES} que incrementa la tasa de error, permanece dentro de los órdenes medidos, aun cuando se utiliza codificación para el control de errores. Sin embargo, y debido al efecto catarata de la curva de tasa de error en bits de los sistemas que utilizan códigos eficientes de decodificación iterativa como los códigos LDPC y los códigos turbo, ese efecto de propagación resulta en una perdida que medida en decibeles, es considerablemente menor y aproximadamente igual a $1\,dB$ en términos del parámetro E_b/N_0.

Así, un sistema que utilize el algoritmo AES, codificación para el control de errores

y permutación aleatoria de los datos encriptados a trasmitir, puede incrementar notablemente su capacidad de encriptamiento, presentando una perdida muy pequeña en su performance de la tasa de error, menor que $1\,dB$, con respecto a un sistema equivalente con codificación para el control de errores, pero sin ningún grado de privacidad. Así en la región de E_b/N_0 entre $2-10\,dB$, la cual es una región muy importante en muchas aplicaciones, y donde el uso del algoritmo AES produce una pérdida en la performance de la tasa de error de alrededor $2-5\,dB$ con respecto a la transmisión sin codificar, los esquemas propuestos reducen esta pérdida a valores menores que $1\,dB$, haciendo uso de códigos LDPC y turbo.

Los esquemas propuesto pueden lograr una performance en la tasa de error de $p_{be} \cong 7 \times 10^{-6}$ con $E_b/N_0 \cong 3dB$, mientras que un esquema que use el algoritmo AES sin codificación para control de errores necesita un $E_b/N_0 \cong 11\,dB$ para lograr la misma performance en la tasa de error.

Los esquemas propuestos no sólo producen una importante ganancia en la performance de la tasa de error, sino también un incremento en los niveles de encriptamiento, mediante la inclusión de simples pero efectivas operaciones de permutación sobre los elementos de la palabra codificada que se va a transmitir.

Debido a su gran nivel de encriptamiento y excelente performance de la tasa de error estos sistemas pueden ser de gran utilidad en aplicaciones donde se requiere una transmisión eficiente y de gran privacidad, como es el caso de las comunicaciones móviles o inalámbricas.

Sin embargo, el efecto de propagación de errores no es totalmente contrarrestado en los esquemas propuestos. Esto se debe a que su reducción se produce por una disminución de los errores de la información que llega al descifrador del receptor, no obstante, si la información a ser desencriptada contiene errores, el efecto de propagación de errores está todavía presente.

El efecto de propagación de errores del algoritmo AES podría ser contrarrestado si la información es primeramente codificada para controlar errores, y luego de algún procedimiento de mezclado de datos, es encriptada por este algoritmo. De esta forma las ráfagas de ruido ocasionadas por el AES en la decodificación, serían desparramadas en varias palabras de código por el proceso de mezclado de datos y luego mejor controladas por el decodificador correspondiente. Sin embargo, y como se explico en la sección 6.2.2, este ordenamiento de los bloques ocasiona un endurecimiento de las decisiones en el decodificador, que le reduce fuertemente su poder de corrección de errores.

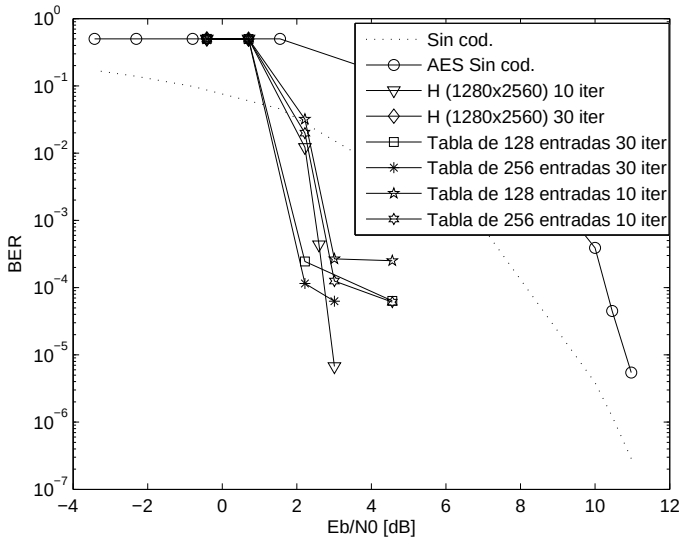

Figura 6.11: BER del AES con código LDPC $C_{LDPC}(2560, 1280)$ para diferentes tamaños de tabla de búsqueda.

6.2.7. Implementación práctica combinando un código LDPC con matriz de chequeo de paridad H de tamaño 2560×1280 y el algoritmo AES de 128 bits

Como se mostró en la sección 5.2.6, es posible implementar un decodificador LDPC [27, 106, 107, 108, 110, 112] en forma muy sencilla en lógica programable utilizando un decodificador de sumas-restas en punto fijo. En este decodificador no se realizan productos ni cocientes, ni se utiliza aritmética de punto flotante. El decodificador propuesto se describe en [105, 106, 107, 108, 109, 110, 111].

Los resultados obtenidos no difieren en gran medida de los obtenidos en forma teórica, cuando se utiliza un decodificador con una tabla de búsqueda de 256 entradas de 2 bytes y 30 iteraciones, tal como se observa en la figura 6.11. En [85, 124, 125, 126] se describe la forma de implementar en lógica programable un encriptador y desencriptador AES de 128 bits de bloques y una llave de 128 bits de bajo costo y área mínima [48, 127]. La combinación de las implementaciones mencionadas permite desarrollar un esquema de transmisión de datos en lógica programable con un alto nivel de privacidad y excelente performance en cuanto a la tasa de error (BER).

6.3. Conclusiones

Como se observó en la sección 6.2.1, la transmisión de datos encriptados por medio del algoritmo AES en presencia de ruido va acompañada de un efecto de propagación de errores proporcional al tamaño de bloque del encriptador.

El incremento del número de errores puede ser obtenido como una multiplicación de la tasa de error (BER) sin codificar por un factor que es aproximadamente la mitad del tamaño del bloque que se utiliza en el proceso de encriptamiento. Desde este punto de vista, hay una relación de compromiso entre la performance de la tasa de error y el grado de encriptamiento.

Esto sugiere el uso de este algoritmo conjuntamente con eficientes técnicas de codificación para el control de errores como los códigos LDPC y turbo, debido a que el requisito de relación señal-ruido es considerablemente alto si se pretende lograr tasas de error aceptables en la mayoría de las aplicaciones prácticas. También es de notar que si desea transmitir información encriptada pero sin estar protegida contra el espectro del ruido, se requieren valores bastante elevados de E_b/N_0 para lograr valores prácticos de tasa de error.

En esta sección se analizó como es posible mejorar la tasa de error (BER) en una transmisión de mensajes cifrados de un sistema que utilice AES mediante el uso de códigos LDPC y turbo. El bloque de encriptamiento utilizado trae como consecuencia un efecto de propagación de error, que en el caso de la transmisión sin codificar, es equivalente a una perdida de relación señal ruido que fluctúa aproximadamente entre $1\,dB$ y $5\,dB$.

El uso de códigos de decodificación iterativa convierte este efecto en una pérdida menor, de aproximadamente $1\,dB$. La utilización de técnicas de control de error con decodificación iterativa conjuntamente con el algoritmo AES permite lograr una tasa de error de aproximadamente 7×10^{-6} con una relación $E_b/N_0 \cong 3\,dB$, cuando para el caso sin codificar se necesita una relación $E_b/N_0 \cong 11\,dB$ para lograr la misma tasa de error. También se muestra la mejora en la privacidad que se puede obtener en el sistema al permutar los elementos de la palabra codificada que se va a transmitir. Esencialmente el factor de mejora en el nivel de privacidad puede ser $(2L)!$ o $(20L)!$, dependiendo del esquema seleccionado.

Por último se muestra como es posible implementar fácilmente este sistema en lógica programable usando un decodificador LDPC que utiliza sólo sumas y restas y un encriptador-desencriptador AES de 128 bits de bloques y una llave de 128 bits. En [128, 129, 130] se describe como implementar el sistema usando un código turbo en vez de un código LDPC.

Capítulo 7

Conclusiones y futuras investigaciones

7.1. Conclusiones

Los códigos de control de error forman parte sustancial de un sistema de comunicaciones de corto alcance. Dependiendo del tipo de aplicación requerida en la transmisión de datos, no siempre es necesario utilizar la técnica de control de error más poderosa o sofisticada.

Por tal motivo, en esta tesis se han abarcado implementaciones que utilizan diferentes estrategias de control de error. Estas incluyen desde los más simples códigos de control de paridad como los códigos de Hamming o cíclicos, pasando por códigos de control de errores de mediana a alta capacidad de corrección como los convolucionales, y finalizando con las técnicas de control de error reconocidas como las más eficaces, y también complejas, como los códigos LDPC.

Al combinar estos códigos de control de error con el algoritmo de encriptamiento AES, se eleva la complejidad de la implementación, pero se obtiene un elevado nivel de privacidad sin deteriorar la performance del sistema frente al ruido.

En este recorrido se han reportado aportes realizados con referencia a tres aspectos de las comunicaciones inalámbricas de corta distancia:

- Sistemas de control de error para enlaces de baja complejidad.
- Sistemas de control de error para enlaces de alta complejidad.
- Sistemas de control de error para enlaces de alta complejidad con alto grado de privacidad.

Una conclusión directa es que cuanto más eficaz es el código de control de errores, más compleja resulta ser su implementación real.

7.1.1. Conclusiones con respectos a sistemas de control de error para enlaces de baja complejidad

En relación con este aspecto, se presentó un enlace de corto alcance para uso médico basado en rayos infrarrojos. Éste es un sistema de telemetría completo, el cual utiliza un código Hamming para la detección y corrección de errores. Luego se demuestra que es posible mejorar la performance del sistema modificando la estrategia de control de error utilizada.

Se describieron los siguientes diseños e implementaciones en lógica programable (FPGA) de ALTERA, de enlaces inalámbricos de baja complejidad:

- Un prototipo de un sistema de telemetría basado en rayos infrarrojos que acoplado a sensores en forma de aguja o lanceta, permite medir diferentes parámetros

fisiológicos. El sistema puede monitorear un máximo de 5 sondas remotas, cada una de las cuales es capaz de manejar hasta 15 parámetros diferentes entregados por los sensores correspondientes.

- Un codificador cíclico $(7, 4)$ para ser utilizado en comunicaciones infrarrojas interiores. Este sistema produce una interesante mejora en el funcionamiento del enlace infrarrojo. La implementación permite facilmente modificar el sistema para poder utilizar cualquier código (n, k) cíclico.

- Un codificador convolucional $(2, 1, 3)$ y su correspondiente decodificador basado en el algoritmo de Viterbi. El codificador convolucional $(2, 1, 3)$ utiliza polinomios generadores $g1 = 101$ y $g2 = 111$. La palabra sin codificar tiene una longitud de 5 bits y la palabra codificada de 10 bits.

Como conclusión de este primer aspecto es que no siempre es necesario utilizar la técnica de control de error más poderosa o sofisticada, para lograr un enlace inalámbrico de corto alcance con una buena performance.

También se demuestra que el uso de radiación infrarroja es una alternativa válida para construir este tipo de enlaces, debido a que se puede disminuir notablemente el consumo del transmisor.

7.1.2. Conclusiones con respectos a sistemas de control de error para enlaces de alta complejidad

En relación a este aspecto, se describe un algoritmo de suma-resta que:

- Solo utiliza operaciones de sumas, restas en punto fijo y búsquedas en tablas, sin necesidad de realizar productos ni cocientes, ni el uso de aritmética de punto flotante.

- Permite trabajar con cualquier tipo de matriz paridad (incluidas las generadas aleatoriamente, las generadas con una determinada regla de construcción, como las quasi-cíclicas, sean de tipo sistemático, o no sistemático), y se adapta en forma paramétrica cualquiera sea la tasa del código k/n. Esto le otorga una amplia versatilidad de aplicación y permite la utilización de los códigos LDPC más eficientes, que son los que utilizan matrices generadas aleatoriamente.

- Posibilita convertir las operaciones de suma, resta, cociente y multiplicación en punto flotante a operaciones de suma-resta en punto fijo, puede ser utilizada no solo para códigos LDPC, sino también por otro tipo de sistemas de decodificación, como por ejemplo los códigos turbo.

- Permite implementar un decodificador LDPC en lógica programable de muy baja complejidad en términos del hardware empleado, con muy buena performance.
- No es necesario medir la densidad espectral de potencia de ruido N_0 del canal para cada símbolo recibido, sino solo realizar una estimación de N_0 del canal periódicamente, reduciendo notablemente la complejidad de la implementación del decodificador.
- El sistema obtenido muestra una gran versatilidad de aplicación y no presenta diferencias apreciables en la performance de tasa de error con respecto al uso del decodificador de suma-producto ideal introducido por D. J. C. MacKay y R. M. Neal
- Un decodificador LDPC que emplee programación lineal presenta una performance similar a la obtenida con un decodificador suma-producto clásico.
- Es posible un implementación práctica de un decodificador que emplee programación lineal, en el cual las operaciones complejas son reemplazadas por comparaciones y búsqueda en tablas.

A lo largo de este segundo aspecto, se demostró como algoritmos que tienen una gran complejidad para ser implementados prácticamente en *hardware*, pueden ser descompuestos en operaciones sencillas, apropiadas para ser realizadas en lógica programable.

También es importante notar que debido a que los códigos de paridad de baja densidad (LDPC) tiene su funcionamiento cercano al límite de Shannon, la metodología de diseño presentada permite implementar en lógica programable enlaces inalámbricos de corto alcance con una performance de calidad de funcionamiento superior.

7.1.3. Conclusiones con respectos a sistemas de control de error para enlaces de alta complejidad con alto grado de privacidad

En relación a este aspecto, es un hecho consumado la creciente demanda e implantación de todo tipo de redes inalámbricas en entornos corporativos, empresariales y en el ámbito familiar. Este tipo de redes ofrecen grandes ventajas frente a las tradicionales redes cableadas, tales como facilidad de instalación, amplia cobertura, movilidad, sencilla ampliación, etc.

Sin embargo, estas ventajas conllevan un contrapartida en forma de problemas de seguridad, debido a que las redes inalámbricas son vulnerables al ataque de cualquiera que conozca como interceptar el enlace.

Como solución se recurre a codificar la información a ser transmitida, pero como se vio esto provoca una fuerte propagación de errores, de manera que cuando se lo utiliza en una transmisión inalámbrica por ejemplo, y frente a la acción del ruido, la performance de tasa de error (BER) se deteriora fuertemente.

Se demostró que al combinar el algoritmo de encriptamiento AES con técnicas de control de errores tales como los códigos LDPC o turbo, se obtiene un sistema con muy buena capacidad de privacidad y excelente performance de la tasa de error (BER).

Particularmente se muestra que:

- La transmisión de datos encriptados por medio del algoritmo AES en presencia de ruido va acompañada de un efecto de propagación de errores proporcional al tamaño de bloque del encriptador.

- El incremento del número de errores puede ser obtenido como una multiplicación de la tasa de error (BER) sin codificar por un factor que es aproximadamente la mitad del tamaño del bloque que se utiliza en el proceso de encriptamiento. Desde este punto de vista, hay una relación de compromiso entre la performance de la tasa de error y el grado de encriptamiento.

- Si se transmite información encriptada pero sin estar protegida contra el espectro del ruido, se requieren valores bastante elevados de E_b/N_0 para lograr valores prácticos de tasa de error.

- Es posible mejorar la tasa de error (BER) en una transmisión de mensajes cifrados de un sistema que utilice AES mediante el uso de códigos LDPC y turbo. El bloque de encriptamiento utilizado trae como consecuencia un efecto de propagación de error, que en el caso de la transmisión sin codificar, es equivalente a una perdida de relación señal ruido que fluctúa aproximadamente entre $1\,dB$ y $5\,dB$. El uso de códigos de decodificación iterativa convierte este efecto en una pérdida menor, de aproximadamente $1\,dB$, respecto del caso no encriptado.

- La utilización de técnicas de control de error con decodificación iterativa conjuntamente con el algoritmo AES permite lograr una tasa de error de aproximadamente 7×10^{-6} con una relación $E_b/N_0 \cong 3\,dB$, cuando para el caso sin codificar se necesita una relación $E_b/N_0 \cong 11\,dB$ para lograr la misma tasa de error.

- Se tiene una mejora en la privacidad del sistema al permutar los elementos de la palabra codificada que se va a transmitir. Esencialmente el factor de mejora en el nivel de privacidad puede ser $(2L)!$ o $(20L)!$, dependiendo del esquema seleccionado.

- Es posible implementar fácilmente este sistema en lógica programable usando un decodificador LDPC que utiliza sólo sumas y restas y un encriptador-desencriptador AES de 128 bits de bloques y una llave de 128 bits.

7.2. Futuras investigaciones

Dentro de la temática de esta tesis un tema de investigación no agotado es el de obtener sistemas de comunicaciones móviles de corto alcance seguros y confiables, factibles de ser implementados en lógica programable.

Dado que para los códigos LDPC y turbo, los métodos originales iterativos de decodificación son más complejos que sus versiones logarítmicas, el uso de algoritmos de decodificación logarítmicos (como por ejemplo el de suma-resta presentado), permite implementar decodificadores de baja complejidad en términos de los recursos de *hardware* empleado.

En base a esta idea se pretende realizar un estudio minucioso de diferentes técnicas de codificación que permitan implementar decodificadores en lógica programable utilizando sumas-restas en punto fijo.

Un aspecto adicional estudiado en la tesis fue la aplicación conjunta de técnicas de control de errores y encriptamiento, orientando estas implementaciones al terreno de las comunicaciones inalámbricas, que hoy día, reclaman un fuerte control de errores unido a una alta necesidad de transmisión con privacia. En este sentido se abre un campo de trabajo extenso relativo a las implicancias de la utilización conjunta de estas técnicas de la Teoría de la Información y la factibilidad de su implementación en dispositivos lógicos programables.

Apéndice A

Algebra logarítmica

Con frecuencia en el algoritmo log-MAP se utilizan expresiones del tipo $\ln(e^a + e^b)$ y $\ln(e^a - e^b)$. Por esta razón, en esta sección se van a desarrollar algunas propiedades de las sumas y restas logarítmicas que se utilizarán en la Tesis.

Dados dos números reales A y B, se los puede representar como $A = e^a$ y $B = e^b$ por tanto su suma C también se la puede representar como $C = e^c$ y está dada por $e^a + e^b$ y su resta por $e^a - e^b$. Ahora se verá que consideraciones hay que tener en cuenta cuando se aplican logaritmos a la suma y resta.

A.1. Suma

A continuación, se presentará una forma sencilla de poder obtener el logaritmo de la suma de dos número exponenciales. La suma $C = e^a + e^b$, se puede poner como:

$$C = e^c = e^a + e^b = (1 + \frac{e^b}{e^a})e^a = (1 + e^{b-a}) \cdot e^a \tag{A.1}$$

como se desea obtener el valor de el logaritmo de la suma, es decir $c = \ln C$; por tal motivo, si se aplican logaritmos a ambos miembros de la ecuación A.1; c resulta ser:

$$c = \ln C = a + \ln(1 + e^{b-a}) \tag{A.2}$$

si $a > b$ entonces $b - a < 0$ se puede poner como $-|b - a|$ quedando entonces:

$$c = a + \ln(1 + e^{-|b-a|}) \tag{A.3}$$

por el contrario si $a < b$ se tiene que $b - a > 0$ obteniendo:

$$c = a + \ln(1 + e^{|b-a|}) \tag{A.4}$$

Si ahora la suma de $A + B$ se la describe como:

$$e^c = e^a + e^b = (1 + \frac{e^a}{e^b}) \cdot e^b = (1 + e^{a-b}) \cdot e^b \tag{A.5}$$

entonces c es igual a:

$$c = b + \ln(1 + e^{a-b}) \tag{A.6}$$

si $a < b$ entonces $a - b < 0$ se puede poner como $-|a - b|$ o $-|b - a|$ quedando:

$$c = b + \ln(1 + e^{-|b-a|}) \tag{A.7}$$

y si $a > b$ se tiene:

$$c = b + \ln(1 + e^{|a-b|}) \tag{A.8}$$

con las ecuaciones A.3 y A.7 se cubren los caso $a > b$ y $a < b$, se puede unir en:

$$c = \text{máx}(a, b) + \ln\left(1 + e^{-|b-a|}\right) \tag{A.9}$$

Si A y B son menores que uno, entonces a y b son negativos. Es decir $a < 0$ y $b < 0$, por tanto C queda:

$$C = e^c = e^{-|a|} + e^{-|b|} \tag{A.10}$$

con:

$$c = -\,\text{mín}(|a|, |b|) + \ln\left(1 + e^{-||b|-|a||}\right) \tag{A.11}$$

Si se tiene que $A + B \leq 1$, entonces se puede poner:

$$C = e^{-|c|} = e^{-|a|} + e^{-|b|} \tag{A.12}$$

en este caso la ecuación A.11 queda:

$$-|c| = -\,\text{mín}(|a|, |b|) + \ln\left(1 + e^{-||b|-|a||}\right) \tag{A.13}$$

resultando:

$$|c| = \text{mín}(|a|, |b|) - \ln\left(1 + e^{-||b|-|a||}\right) \tag{A.14}$$

Ahora se define un factor de corrección $f_+(|a|, |b|)$ como:

$$f_+(|a|, |b|) = \ln\left(1 + e^{-||b|-|a||}\right) \tag{A.15}$$

entonces de la ecuación A.12 se tiene:

$$\ln C = \ln(e^{-|a|} + e^{-|b|}) = -|c| \tag{A.16}$$

siendo $|c|$:

$$|c| = \text{mín}(|a|, |b|) - f_+(|a|, |b|) \tag{A.17}$$

De las ecuaciones A.12, A.16 y A.17 se ve que para realizar una suma logarítmica, en la cual $A + B \leq 1$ sólo es necesario realizar una operación de comparación entre los exponentes $|a|$ y $|b|$, mientras que el factor de corrección $f_+(|a|, |b|)$ puede estar almacenado en una memoria o tabla de búsqueda cuyas entradas sean $||b| - |a||$.

A.2. Resta

En el caso de desear obtener el logaritmo de la resta $C = A - B$, el problema se puede dividir en dos casos: $C > 0$ si $A > B$ y $C < 0$ si $A < B$. Definiendo $|C| = |A - B|$, la resta se puede escribir como:

$$C = (-1)^n \cdot |C| \quad \textit{donde} \quad n = 0 \quad \textit{si} \quad A > B \quad \textit{o} \quad n = 1 \quad \textit{si} \quad A < B \tag{A.18}$$

Primer caso: $A > B$ (con $A = e^a$ y $B = e^b$), entonces $a > b$:

$$e^c = e^a - e^b = (1 - \frac{e^b}{e^a}) \cdot e^a = (1 - e^{b-a}) \cdot e^a \tag{A.19}$$

si se aplican logaritmos, c resulta ser igual a:

$$c = a + \ln(1 - e^{b-a}) \tag{A.20}$$

si $a > b$ entonces $b - a < 0$ se puede poner como $-|b - a|$ quedando entonces:

$$c = a + \ln(1 - e^{-|b-a|}) \tag{A.21}$$

Segundo caso: $A < B$ entonces $a < b$:

$$C = (-1)e^c = e^a - e^b = (-1) \cdot (e^b - e^a) \tag{A.22}$$

si:

$$e^c = (e^b - e^a) = (1 - \frac{e^a}{e^b}) \cdot e^b = (1 - e^{a-b}) \cdot e^b \tag{A.23}$$

aplicando logaritmos, c es igual a:

$$c = b + \ln(1 - e^{a-b}) \tag{A.24}$$

si $a < b$ entonces $a - b < 0$ se puede poner como $-|a - b|$ ó $-|b - a|$ quedando entonces:

$$c = b - \ln(1 - e^{-|b-a|}) \tag{A.25}$$

Al igual que en el caso de la suma, con las ecuaciones A.21 y A.25 se cubren los caso $a > b$ y $b > a$. Uniendo ambos casos, se obtiene:

$$c = \text{máx}(a, b) + \ln(1 - e^{-|b-a|}) \tag{A.26}$$

Si A, B son menores que uno, C también lo será. En este caso a, b y c son negativos, (es decir $a < 0$, $b < 0$ y $c < 0$), la resta resulta ser:

$$C = (-1)^n \cdot e^{-|c|} = (-1)^n \cdot |e^{-|a|} - e^{-|b|}| \tag{A.27}$$

si $c < 0$, la ecuación A.26 se puede escribir como:

$$-|c| = -\,\text{mín}(|a|, |b|) + \ln(1 - e^{-||b|-|a||}) \tag{A.28}$$

entonces:

$$|c| = \text{mín}(|a|, |b|) - \ln(1 - e^{-||b|-|a||}) \tag{A.29}$$

como $(1 - e^{-||b|-|a||}) < 1$ resulta $\ln(1 - e^{-||b|-|a||}) < 0$, entonces se puede poner la ecuación A.29 como:

$$|c| = \text{mín}(|a|, |b|) + |\ln(1 - e^{-||b|-|a||})| \tag{A.30}$$

Si ahora se define el factor de corrección $f_-(|a|, |b|)$ como:

$$f_-(|a|, |b|) = |\ln(1 - e^{-||b|-|a||})| \tag{A.31}$$

entonces queda:

$$|c| = \text{mín}(|a|, |b|) + f_-(|a|, |b|) \tag{A.32}$$

Por tanto la resta de $C = A - B = e^{-|a|} - e^{-|b|}$ se puede escribir como:

$$C = (-1)^n \cdot e^{-|c|} \tag{A.33}$$

donde $n = 0$ si $A > B$ es decir si $|a| < |b|$ y $n = 1$ si $A < B$ o sea si $|a| > |b|$

De las ecuaciones A.32 y A.33 se ve que para realizar una resta logarítmica, sólo es necesario realizar una operación de comparación entre los exponentes $|a|$ y $|b|$, mientras que el factor de corrección $f_-(|a|, |b|)$ puede estar almacenado en una memoria o tabla de búsqueda cuyas entradas sean $||b| - |a||$.

A.3. Cociente

Para el caso de los cocientes se tiene:

$$C = \frac{A}{A + B} \tag{A.34}$$

y si se supone que $A < 1$ y $B < 1$, entonces $C < 1$ queda:

$$e^{-|c|} = \frac{e^{-|a|}}{e^{-|a|} + e^{-|b|}} \tag{A.35}$$

aplicando logaritmo a ambos miembros de la ecuación A.34:

$$\ln(C) = \ln(A) - \ln(A + B) \tag{A.36}$$

y utilizando la ecuación A.35:

$$-|c| = -|a| - \ln(e^{-|a|} + e^{-|b|}) \tag{A.37}$$

es decir:

$$|c| = |a| + \ln(e^{-|a|} + e^{-|b|}) \tag{A.38}$$

y aplicando a la ecuación A.38, lo obtenido en las ecuaciones A.16 y A.17, se tiene:

$$|c| = |a| - \text{mín}(|a|, |b|) + f_+(|a|, |b|) \tag{A.39}$$

De nuevo se ve que para realizar el cociente en forma logarítmica de dos números solo es necesario realizar una operación de comparación y aplicar un factor de corrección.

A.4. Resumen de operaciones

En la tabla A.1 se presenta un resumen de las operaciones logarítmicas:

Suma	$\ln(e^{-\lvert c\rvert}) = \ln(e^{-\lvert a\rvert} + e^{-\lvert b\rvert}) = -\lvert c\rvert$	$\lvert c\rvert = \text{mín}(\lvert a\rvert, \lvert b\rvert) - f_+(\lvert a\rvert, \lvert b\rvert)$
Resta	$\ln(e^{-\lvert c\rvert}) = \ln(e^{-\lvert a\rvert} - e^{-\lvert b\rvert}) = -\lvert c\rvert$	$\lvert c\rvert = \text{mín}(\lvert a\rvert, \lvert b\rvert) + f_-(\lvert a\rvert, \lvert b\rvert)$
Cociente	$\ln(e^{-\lvert c\rvert}) = \ln(\frac{e^{-\lvert a\rvert}}{e^{-\lvert a\rvert}+e^{-\lvert b\rvert}}) = -\lvert c\rvert$	$\lvert c\rvert = \lvert a\rvert - \text{mín}(\lvert a\rvert, \lvert b\rvert) + f_+(\lvert a\rvert, \lvert b\rvert)$

Tabla A.1: Resumen de las operaciones logarítmicas

Apéndice B

Estructura de los dispositivos lógicos programables

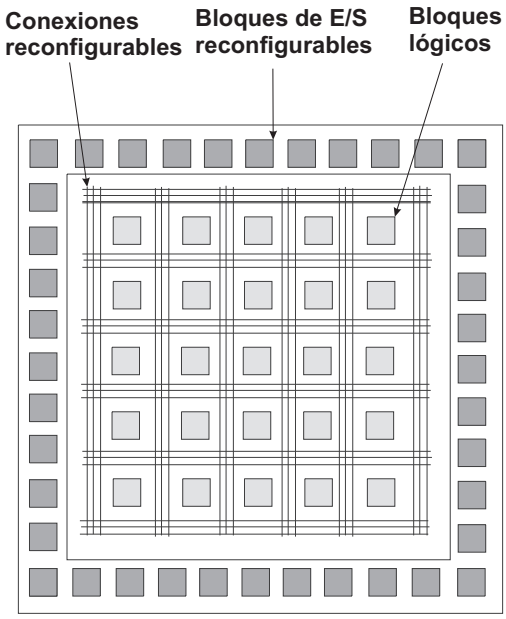

Figura B.1: Estructura general de un dispositivo lógico programable.

En la figura B.1 se muestra un esquema general de la estructura de los dispositivos lógicos programables [81]. En esta visión se utilizan los siguientes conceptos globales:

- Funciones lógicas reconfigurables (celdas lógicas)
- Elementos de entrada/salida reconfigurables.
- Interconexiones reconfigurables.
- Memoria de configuración.

B.1. Funciones lógicas reconfigurables

Todos los dispositivos lógicos programables contienen celdas lógicas básicas programables replicadas en forma matricial dentro del circuito integrado. Los diferentes tipos de celdas lógicas basicas utilizadas son:

- Celdas basadas en multiplexores.
- Celdas basadas en tablas de búsqueda, que en inglés se conocen como *Look-Up Tables* (LUT).
- Celdas basadas en suma de términos producto.

Una celda lógica es un bloque funcional capaz de resolver una función combinatoria o secuencial. Una celda que por sí sola puede llegar a resolver una función compleja (4 o más variables de entrada) genera circuitos de baja granularidad (lógica de grano grueso), mientras que una celda capaz de resolver únicamente funciones elementales de 2 o 3 variables (a veces sólo combinatorias) es llamada de grano fino.

Cuanto más compleja es una celda, mayor es la posibilidad de su subempleo. Sin embargo, en este caso la mayor parte del problema es resuelto dentro de la celda, y por ello se requieren menos recursos de interconexión con otras celdas. En contrapartida cuanto más simples son las celdas: se requieren varias para resolver un problema dado, suelen ser usadas plenamente (el subempleo de celdas es menor) y se requiere de mayores recursos de interconexión debido a la necesidad de múltiples vínculos entre celdas.

La conectividad entre celdas requiere el uso de vínculos que ocupan área del chip. Cuanto más larga es una conexión, mayor es su inductancia, su resistencia y su capacidad parásita, y con ello los retardos de propagación que introduce. Una conexión corta, en contrapartida obliga a que las dos celdas que ella conecta estén físicamente próximas entre sí dentro del chip. Todos los fabricantes ofrecen por ello jerarquías de conectividad, de distinto alcance [82, 83, 131].

B.2. Elementos de entrada/salida reconfigurables

Las celdas de entrada/salida (I/O) son las responsables de la interconexión del dispositivo con el mundo externo, deben manejar corrientes importantes y capacidades parásitas decenas de veces superiores a las existentes dentro del *chip*; además deben tener la capacidad de ser bidireccionales. Contribuyen fuertemente a crear ciertos problemas de diseño tales como el agregado de retardos, la generación de picos de corriente de consumo y el consiguiente ruido de fuente.

B.3. Interconexiones reconfigurables

Las interconexiones en los dispositivos lógicos programables están muy relacionadas con la granularidad, es decir cuanto menor es la granularidad (celdas menos complejas) más recursos de cableado se requieren para resolver la misma función. Las conexiones poseen estructuras jerárquicas teniendo en cuenta su alcance y se pueden agrupar en:

- Locales: son conexiones que unen cada celda con sus celdas vecinas.
- Vecinales: permiten conectar grupos de celdas cercanas.

- Globales: permiten conectar cualquier celda con cualquier otra dentro del dispositivo.

B.4. Memoria de configuración

Los dispositivos lógicos programables, como se muestra en la figura B.1, poseen funciones lógicas, interconexiones y elementos de entrada/salida programables o configurables. Dicha programación es realizada por una cadena de bits creada por un *software* especial e ingresada al circuito integrado por las entradas asignadas a tal efecto. A grandes rasgos las distintas tecnologías de programación se pueden agrupar en:

- Antifusibles: dispositivos normalmente abiertos que una vez programados pasan a ser un cortocircuito, en un proceso irreversible [78, 131, 132, 133].
- Llaves EPROM/EEPROM: dispositivos de paso reprogramables, cuyo estado no se altera al cortarse la alimentación de energía [133].
- Llaves SRAM: basados en llaves de paso MOS controladas desde un *flip flop* estático, cuyo contenido se borra al cortarse la energía [133].

Apéndice C

Estilos de descripción en VHDL

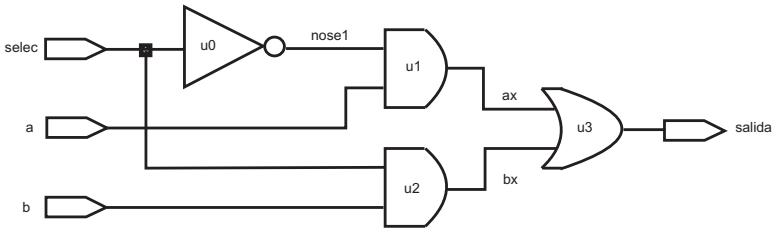

Figura C.1: Esquema de un ejemplo básico en VHDL.

VHDL presenta tres estilos de descripción de circuitos dependiendo del nivel de abstracción. Mediante el ejemplo [75] que se observa en la figura C.1 se va a mostrar la forma en que se escriben descripciones en VHDL. Primero se va a realizar una descripción del comportamiento en forma algorítmica, luego en forma de flujo de datos y por último una descripción estructural. En primer lugar, independientemente de la descripción que se haga (algorítmica, flujo de datos o estructural), hay que definir el símbolo o entidad (ENTITY) del circuito. Es decir, se define una caja negra con las entradas y salidas que posee y con el nombre por el cual se la va a reconocer. En el ejemplo de la figura C.1:

```
ENTITY mux IS
    PORT (
            a: IN bit;
            b: IN bit;
            selec: IN bit;
            salida: OUT bit;
    );
END mux
```

El párrafo anterior indica que la entidad mux (que es el nombre que se le ha dado al circuito) tiene tres entradas de tipo bit y una salida también del tipo bit. El tipo *bit* simplemente indica que la línea puede tomar los valores $'0'$ o $'1'$.

La entidad de un circuito es única, Sin embargo una entidad puede tener diferentes arquitecturas (ARCHITECTURE), en donde se describe el funcionamiento del circuito.

C.1. Descripción algorítmica

A continuación se describe el comportamiento de la entidad mux en forma algorítmica:

```
ARCHITECTURE algorítmica OF mux IS
```

```
BEGIN
    PROCESS(a,b,selec)
    BEGIN
        IF (selec = '0') THEN
            salida <= a;
        ELSE
            salida <= b;
        END IF;
    END PROCESS;
END algorítmica
```

El bloque PROCESS es una especie de subrutina cuyas instrucciones se ejecutarán secuencialmente cada vez que alguna de las señales (a, b, selec) cambie. Este tipo de descripción sigue una estructura parecida a los lenguajes de programación tradicionales, es por eso que se lo denomina descripción de comportamiento algorítmica. Lo que indica es que si la señal selec es cero, entonces la salida toma el valor presente en la entrada *a*, en caso contrario toma el valor presente en la entrada *b*. Nótese que aquí sólo se describe el comportamiento, es decir no se indican los componentes ni sus interconexiones.

C.2. Descripción flujo de datos

La descripción anterior describe solamente el comportamiento mediante una secuencia sencilla de instrucciones. Mediante la descripción flujo de datos es posible describir el circuito de una forma que está más cercana a una posible realización física.

```
ARCHITECTURE flujo OF mux IS
SIGNAL nosel, ax,bx :bit
BEGIN
    nosel <= NOT selec
    ax <= a AND nosel
    bx <= b AND selec
    salida <= ax OR bx
END flujo
```

En esta descripción hay varias instrucciones que se ejecutan cada vez que cambia alguna de las señales que interviene en la asignación.

C.3. Descripción estructural

En este caso se utiliza al VHDL como lenguaje de descripción de estructuras o *netlist* [74, 75].

```
ARCHITECTURE estructura OF mux IS
SIGNAL nosel, ax,bx :bit
BEGIN
    u0: ENTITY inv PORT MAP(e => selec, y => nosel)
    u1: ENTITY and2 PORT MAP(e1 => a, e2 => nosel, y => ax)
    u2: ENTITY and2 PORT MAP(e1 => b, e2 => selec, y => bx)
    u3: ENTITY or2 PORT MAP(e1 => ax, e2 => bx, y => salida)
END estructura
```

En este caso en el cuerpo de la arquitectura se ponen los componentes y sus interconexiones. Para los componentes se utilizan entidades (ENTITY) que están definidas en alguna biblioteca auxiliar. Para las interconexiones se usan señales (SIGNAL) declaradas al comienzo de la arquitectura.

Bibliografía

[1] Altera Corporation. *Data Book*. 2001.

[2] J. Feldman. *Decoding Error-Correcting Codes vía Linear Programming*. PhD thesis, Massachusetts Institute of Techology, USA, 2003.

[3] B. Sklar. *Digital Communications - Fundamentals and Applicatins*. Prentice Hall International, 1988. 0132119390.

[4] J. G. Proakis; M. Salehi. *Communication Systems Engineering*. Prentice Hall, 1994. ISBN 0131589326.

[5] B. Carlson. *Communication Systems-An Introduction To Signals And Noise In Electrical Communication*. Mc. Graw Hill, 1986. ISBN 0071005609.

[6] C. E. Shannon. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27:379–423, 1948.

[7] C. E. Shannon. Communications in the presence of noise. *Proc. of the IEEE*, 86:447–458, 1998.

[8] S. Lin; D.J. Costello. *Error Control Coding: Fundamentals and Applications*. Prentice Hall, Englewood Cliffs, New Jersey, 2004. ISBN 0130426725.

[9] J. Castiñeira Moreira; P. G. Farrell. *Essential off Error-Control Coding*. John Wiley and Sons, England, 2006. ISBN 139780470029206.

[10] R.W. Hamming. Error detecting and error correcting codes. *Bell. Syst. Tech. J.*, 29:147–160, 1950.

[11] M. Blaum. A course on error correcting codes. IBM Research Division, 1997.

[12] W.W. Peterson; E.J. Wledon. *Error-Correcting Codes*. MIT Press, Cambridge, Massachusetts, 1972. ISBN 0262160390.

[13] A.J. Viterbi; J.K. Omura. *Principles of Digital Communication and Coding.* McGraw-Hill, New York, 1979. 0070675163.

[14] A.J. Viterbi. Error bounds for convolutional codes and an asymplotically optimum decoding algorithm. *IEEE Tranas. Inf. Theory*, 13:260–269, 1967.

[15] J.K. Omura. On the Viterbi decoding algorithm. *IEEE Tranas. Inf. Theory*, 15:177–179, 1969.

[16] J. J. Buchholz. Matlab implementation of the advanced encryption standard. http://buchholz.hs-bremen.de/aes/aes.htm.

[17] D. J. C. MacKay. Good error-correcting codes based on very sparse matrices. *Transactions on Information Theory*, 45:399–431, 1999.

[18] D. J. C. MacKay; R. M. Neal. Near Shannon limit performance of low density parity check codes. *Electronics Letters*, 33:457–458, 1997.

[19] R. G. Gallager. Low density parity check codes. *IRE Trans. Information Theory*, IT-8:21–28, 1962.

[20] L.M. Tanner. A recursive approach to low complexity codes. *IEEE Trans. Inf. Theory*, 27:533–547, 1981.

[21] M.C.Davey. *Error-Correction Using Low-Density Parity-Check Codes.* PhD thesis, University of Cambridge, Cambridge, UK, 1999.

[22] H. Tang; J. Xu; Y. Kou; S. Lin; K. Abdel-Ghaffar. On algebraic construction of gallager and circulant low-density parity-check codes. *IEEE Trans. Inform. Theory*, 50:1269–1279, 2004.

[23] B. Ammar; B. Honary; Y. Kou; J. Xu; S. Lin. Construction of low-density parity-check codes based on balanced incomplete block designs. *IEEE Trans. Inform. Theory*, 50:1257–1269, 2004.

[24] Y.Kou; S. Lin; M. Fossorier. Low-density parity-check codes based on finite geometries: A rediscovery and new results. *IEEE Trans. Inform. Theory*, 47:2711–2736, 2001.

[25] F.R. Kschischang; B.J. Frey; H.A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions Information Theory*, 47:498–519, 2001.

[26] A. Anastasopoulos. A comparison between the sum-product and the min-sum iterative detection algorithms based on density evolution. *GLOBECOM 01Global Telecommunications Conference. IEEE*, 2:1021–1025, 2001.

[27] L. Ping; W. K. Leung. Decoding low density parity check codes with finite quantization bits. *IEEE Communications Letters*, 4:62–64, 2000.

[28] J. Zhao; F. Zarkeshvari; A.H. Banihashemi. On implementation of min-sum algorithm and its modifications for decoding low density parity check LDPC codes. *IEEE Transactions on Communications*, 53:549–554, 2005.

[29] C. Howland; A. Blanksby. Parallel decoding architectures for low density parity check codes. *Proc. IEEE Int. Symp. on Circuits and Systems. Sydney*, 4:742–745, 2001.

[30] M.M Mansour;Ñ.R. Shanbhag. Low-power VLSI decoder architectures for LDPC codes. *Low Power Electronics and Design, 2002. ISLPED '02*, pages 284–289, 2002.

[31] T Zhang; K.K. Parhi. VLSI implementation oriented (3,k) regular low density parity check codes. *IEEE Workshop on Signal Processing Systems SIPS. Available at http://www.ecse.rpi.edu/homepages/tzang/*, pages 25–36, 2001.

[32] L. Yang; M. Sheng; H. Liu; C.J. Shi. An FPGA implementation of low density parity check code decoder with multi-rate capability. *IEEE ASP-DAC 2005*, pages 760–763, 2005.

[33] J. Sha; M. Gao; Z. Zhang; L. Li; Z.Wang. A memory efficient FPGA implementation of quasi-cyclic LDPC Decoder. *Proc. 5th WSEAS Int. Conf. on Instrumentation, Measurement, Circuits and Systems*, 1:218–223, 2006.

[34] J. Chen; A. Dholakia; E. Eleftheriou; M. Fossorier; X. Yu Hu. Near optimal reduced-complexity decoding algorithms for LDPC codes. *ISIT 2002, Lausanne, Switzerland*, page 455, 2002.

[35] X. Yu Hu; E. Eleftheriou; D.M. Arnold; A. Dholakia. Efficient implementations of the sum-product algorithm for decoding LDPC codes. *GLOBECOM 01 Global Telecommunications Conference. IEEE*, 2:1036–1036, 2001.

[36] J. Chen; A. Dholakia; E. Eleftheriou; M. Fossorier; X. Yu Hu. Reduced-complexity decoding of LDPC codes. *IEEE Transactions on Communications*, 53:1288–1299, 2005.

[37] M.P.C. Fossorier; M. Mihaljevic; H. Imai. Reduced complexity iterative decoding of low-density parity check codes based on belif propagation. *IEEE Transaction on Communications*, 47:643–679, 1999.

[38] C. Berrou; A. Glavieux; P. Thitimajshima. Near Shannon limit error-correcting coding an decoding: Turbo codes. *IEEE Internacional Conference on Communications*, pages 1064–1070, 1993.

[39] L. Hanzo; T. H. Liew; B. L. Yeap. *Turbo Coding, Turbo Equalisation and Space-Time Coding, for Transmisión over Fading Channels*. IEEE/Wiley, USA, 2002. 0470847263.

[40] C. Heegard; S. Wicker. *Turbo Coding*. Kluver, Massachussets, 1999. ISBN 0792383788.

[41] M.C. Valenti. Turbo codes and iterative processing. *Proc. of the IEEE Int. Conf. on Communications. Geneva, Switzerland*, 2:944–968, 1993.

[42] L. Bahl; J. Cocke; F. Jelinek; J. Raviv. Optimal decoding of linear codes for minimising symbol error rates. *IEEE Trans. Inf. Theory*, IT 20:284–287, 1974.

[43] D. Divsalar; H. Jin; R.J. MacElice. Coding theorems for turbo-like´codes. *Proc. 36th Allerton Conf. Communications, Control and Computing. Urbana, IL*, pages 201–210, 1998.

[44] J.B. Cain; G.C. Clark; J.M. Geist. Punctured convolutional codes of rate (n-1)/n and simplified maximum likelihood decoding. *IEEE Trans. Inf. Theory*, 25:97–100, 1979.

[45] J.P. Woodard; L. Hanzo. Comparative study of turbo decoding techniques. *IEEE Transaction on Vehicular Technology*, 49(6):2208–2233, 2000.

[46] R.J. MacElice; D.J.C. MacKay; J.F. Cheng. Turbo decoding as an instance of Pearl´s belief propagation algorithm. *IEEE J. Select. Areas Commun.*, 16:140–152, 1998.

[47] J. Daemen; V. Rijmen. Aes proposal: Rijndael. document version 2. IRE Trans. Information Theory. NIST´s AES home page, http://www.nist.gov/aes, 1999.

[48] A. Elbirt; W. Yip; B. Chetwynd; C. Paar. An FPGA implementation and performance evaluation of the AES block cipher candidates algorithm finalists. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pages 545–557, 2001.

[49] N. Koblitz. *A Course in Number Theory and Cryptography*. Springer - Verlag, 1994. ISBN 0387942939.

[50] E. Castillo; A.J. Conejo; P. Pedregal; R. García; M. Alguacil. *Building and Solving Mathematical Programming Models in Engineering and Science*. Wiley, New York, USA, 2002. 9780471150435.

[51] J.Feldman; M. J. Wainwright; D. R. Karger. Using linear programming to decode binary linear codes. *IEEE Transaction on Information Theory*, 51.

[52] J. Feldman. *Decoding Error-Correcting Codes via Linear Programming*. PhD thesis, Massachusetts Institute of Technology, Massachusetts, EEUU, September 2003.

[53] J. Feldman; C. Stein. LP decoding achieves capacity. *Proc. Symp. Discrete Aalgorithms SODA05. Vancouver, Canada*, pages 460–469, 2005.

[54] J. M. Kahn; J. R. Barry. Wireless infrared communications. *Proceedings of the IEEE*, 85:265–297, 1997.

[55] A. Tavares; R. Valadas; A. Duarte. Performance of wireless infrared transmission systems considering both ambient light interference and intersymbol interference due to multipath dispersion. *Proceedings of the SPIE*, 3532:82–93, 1995.

[56] Infolink. User Reference. Low Power Radio Solutions LTD.

[57] F.J. López-Hernández; M.J. Betancor. DUSTIN: algorithm for calculation of impulse response on IR wireless indoor channels. *Electronics Letters*, 33:1804–1806, 1997.

[58] M.D. Audeh; J.M. Kahn. Performance evaluation of baseband OOK for wireless indoor infrared LAN´s operating at 100Mb/s. *IEEE Transactions on communications*, 43:2085–2094, 1995.

[59] A.C. Boucouvalas; P. Barker. Asymetry in optical wireless link. *IEE Proc. Optoelectron.*, 147:315–321, 2000.

[60] K. Samaras; A.M. Street; D.C. O´Brien; D.J. Edwards. Method for evaluating error rates in infrared wireless links. *Electronics Letters*, 33:1720–1721, 1997.

[61] H. Park. Power efficient coded modulation and precoding schemes for wireless infrared communications. *IEEE*, pages 614–618, 1999.

[62] J.B. Carruthers; J.M. Kahn. Modeling of nondirected wireless infrared channels. *IEEE Trans. Commun.*, 45:1260–1268, 1997.

[63] J.M. Kahn; W.J. Krause; J.B. Carruthers. Experimental characterization of non-directed indoor infrared links. *IEEE Trans. Commun.*, 43:1613–1623, 1995.

[64] A.J.C. Moreira; R.T. Valadas; A.M. Duarte. Performance of infrared transmission systems under ambient ligth interference. *IEE Proc. J.*, 143:339–346, 1996.

[65] A.J.C. Moreira; M.R. Tavares; R.T. Valadas; A.M. Duarte. Modulation methods for wireless infrared transmission systems performance under ambient light noise and interference. *Proceedings SPIE*, 2601:226–237, 1995.

[66] H. Park; J. Barry. Modulation analysis for wireless infrared communications. *IEEE International Conference on Communications, ICC´95*, 2:1182–1186, 1995.

[67] G. W. Marsh; J. M. Kahn. Channel reuse strategies for indoor infrared wireless communications. *IEEE Transaction on Communications*, 45:1280–1290, 1997.

[68] Z. Ghassemlooy; A.R. Hayes; N. L. Seed. Digital pulse modulation for optical communications. *IEEE Comm. Mag.*, 36:95–99, 1998.

[69] E. Boemo; F. Barbero Diaz; G. González; E. Juarez. *Introducción al diseño con LCAs*. Servicio de Publicaciones E.T.S.I. Telecomunicación, 1993. ISBN 84-7402-233-9.

[70] Monolithic Memories. *Programmable Logic Array Handbook*. 1986.

[71] W. Carter; K. Duong; R. Freeman; H. Hsieh; J. Ja; J. Mahoney; L. Ngo; S Sze. A User Programmable Reconfigurable Logic Array. *Proceedings of the IEEE Custom Integrated Circuits Conference*, pages 233–235, 1986.

[72] S. Brown; J. Rose. FPGA and CPLD Architectures: a Tutorial. *IEEE Design and Test of Computers*, 13(2):42–57, 1996.

[73] T. Riesgo; Y. Torroja; E. de la Torre. Design Metodologies on Hardware Description Languages. *IEEE Transactions on Industrial Electronics*, 46(1):3–12, 1999.

[74] L. Terés; Y. Torroja; S. Olcoz; E. Villar. *VHDL. Lenguaje Estándar de Diseño Electrónico*. McGraw Hill/Interamerican de España, 1997. ISBN 8448111966.

[75] F. Pardo; J. Boluda. *VHDL. Lenguaje para Síntesis y Modelado de Circuitos*. RA-MA, 1999. ISBN 8478973516.

[76] A. Ye. *Field-Programmable Gate Array Architectures and Algorithms Optimized for Implementing Data Path Circuits*. PhD thesis, University of Toronto, Toronto, Ontario Canada, November 2004.

[77] E. Boemo; G. Sutter; E. Todorovich; S. López-Buedo. *FPGAs Based Systems*. School of Computer Engineering, UAM, 2006. ISBN 8460989984.

[78] M. J. S. Smith. *Application-Specific Integrated Circuits*. Addison-Wesley, 1999. ISBN 0201500221.

[79] S. Brown; R. Francis; J. Rose; Z. Vranesic. *Field-Programmable Gate Arrays*. Kluwer Academic Publisher, 1992.ISBN 9780792392484.

[80] Y. Torroja; T. Riesgo; J. Uceda. El proceso de diseño basado en lenguajes de descripción de hardware. *Novatica*, pages 54–67, 1995.

[81] C. Gayoso; C. M. González. Apuntes de Cátedra. Diseño Digital con Técnicas de Alto Nivel. Facultad de Ingeniería, Universidad Nacional de Mar del Plata. 2000.

[82] www.xilinx.com. *On Line*.

[83] www.altera.com. *On Line*.

[84] Altera Corporation. *MAX + PLUS II. VHDL*. 1996.

[85] M. Liberatori. *Desarrollo de Encriptado AES en FPGA. Implementación de Area Mínima y Bajo Costo*. PhD thesis, Universidad Nacional de La Plata, La Plata, Argentina, Abril 2006.

[86] F. Carden. *Telemetry Systems Design*. Artech House, INC., 1995. ISBN 0890068003.

[87] J. R. Barry; J. M. Kahn; W. J. Krause; E. A. Lee; D. G. Messerschmitt. Simulation of multipath impulse response for wireless optical channels. *IEEE J. Select Areas in Communication*, 11:367–379, 1993.

[88] A. Tavares; R. Valadas; A. Duarte. Performance of an optical sectored receiver for indoor wireless communication system in presence of artificial and natural noice sources. *Proceedings of the SPIE 2601*, pages 264–273, 1995.

[89] R. Valadas; A. Tavares; A. Duarte. Angle diversity to combat the ambient noise in indoor optical wireless communications systems. *IJWIN International Journal of Wireless Information Networks*, 4:275–288, 1997.

[90] L. Arnone; C. Gayoso; C. González; A. Ivorra; D. Marín. Sistema de transmisión para uso médico basado en rayos infrarrojos. *VII Workshop de Iberchip. Montevideo, Uruguay*, page 15, 2001.

[91] L. Arnone; C. Gayoso; C. González; A. Ivorra; D. Marín. Aplicaciones de la microelectrónica en biotelemetría. *XIII Congreso Argentino de Bioingeniería, II Jornadas de Ingeniería Clínica. SABI 2001. Tucumán, Argentina*, page 147, 2001.

[92] C. Gayoso; C. González; L. Arnone; J.C. García. Desarrollo de un sistema integral de biotelemetría: Monitorización remota de señales biológicas. *VIII Workshop de Iberchip. Guadalajara, México*, page 50, 2002.

[93] F. R. Gfeller; U. H. Bapst. Wireless in house data communication via diffuse infrared radiation. *IEEE Proceedings*, 67:1474–1486, 1979.

[94] P. Pallás Areny. *Transductores y acondicionadores de señal*. Marcombo, 1989. ISBN 8426707645.

[95] R. Ananth; M Noll; K. Phang. Low voltage infrared transceiver design. IBM Microelectronics.

[96] IBM31T1100A. Application notes controller interface. IBM.

[97] W. Hirt. IrDA-VFlr 16Mb/s: Modulation code and system design. *IEEE Personal Communications*, pages 58–71, 2001.

[98] J.R. Pierce. Optical channels: Practical limits with photon counting. *IEEE Trans. Commun.*, 26:1819–1821, 1978.

[99] G. W. Marsh; J. M. Kahn. Performance evaluation of experimental 50Mb/s diffuse infrared wireless link using on-off keying with decision freedback equalization. *IEEE Transaction on Communications*, 44:1496–1504, 1996.

[100] HSDL-1100 Infrared transceiver. Technical data. HEWLETT PACKARD.

[101] L. Arnone; C. Gayoso; C. González; J.C. García; J. Castiñeira Moreira. Diseño de un codificador cíclico programable para ser utilizado en comunicaciones infrarrojas interiores. *VIII Congreso Argentino de Ciencias de la Computación. Buenos Aires, Argentina*, page 30, 2002.

[102] L. Arnone; C. Gayoso; C. González; J.C. Garcia; J. Castiñeira Moreira. Implementación en lógica programable de un codificador cíclico para enlaces infrarrojos. *III Jornadas sobre Computación Reconfigurable y Aplicaciones. Madrid, España*, pages 283–290, 2003.

[103] L. Arnone; C. Gayoso; C. González; J.C. Garcia; J. Castiñeira Moreira. Diseño de un decodificador Viterbi para ser utilizado en enlaces infrarrojos. *IV Workshop de Iberchip. La Habana, Cuba*, page 43, 2003.

[104] L. Arnone; C. Gayoso; C. González; J. Castiñeira Moreira. Logic programmable implemetation of convolutional coding for indoors infrared links. *X RPIC, X Reunión de Trabajo en Procesamiento de la Información y Control. San Nicolás, Argentina*, pages 781–785, 2003.

[105] L. Arnone; C. Gayoso; C. González; J. Castiñeira Moreira. Implementación de códigos de paridad de baja densidad en lógica programable usando sumas y restas. *IV Jornadas sobre Computación Reconfigurable y Aplicaciones. Barcelona, España*, pages 431–439, 2004.

[106] L. Arnone; C. Gayoso; C. González; J. Castiñeira Moreira. A LDPC logarithmic decoder implementation. *VIII International Symposium on Communications Theory and Applications. Ambleside,UK*, pages 356–361, 2005.

[107] L. Arnone; C. Gayoso; C. González; J. Castiñeira Moreira. Algoritmo de suma-resta para implementar códigos de paridad de baja densidad en lógica programable. *XI Workshop de IBERCHIP. Salvador Bahia, Brazil*, pages 127–129, 2005.

[108] L. Arnone; C. Gayoso; C. González; J. Castiñeira Moreira. Sum-subtract fixed point ldpc decoder. *Latin American Applied Research*, 37:17–20, 2007.

[109] L. Arnone; C. Gayoso; C. González; J. Castiñeira Moreira. Sum-subtract fixed point ldpc logarithmic decoder. *XI RPIC, XI Reunión de Trabajo en Procesamiento de la Información y Control Río Cuarto, Argentina*, pages 8–9, 2005.

[110] L. Arnone; C. Gayoso; C. González; J. Castiñeira Moreira. Sum-subtract fixed point LDPC decoder. *SPL06, II Southern Conference on Programmable Logic. Mar del Plata, Argentina*, pages 155–161, 2006.

[111] L. Arnone; C. González; C. Gayoso; J. Castiñeira Moreira. Implementación paramétrica en FPGA de un decodificador LDPC para cualquier tipo de matriz paridad y tasa de código. *Aceptado para publicar en el XIV Workshop de IBERCHIP. Puebla. México*, 2008.

[112] T. Bhatt; K. Narayanan; M. Kehtarnavaz. Fixed point DSP implementation of Low-Density Parity Check Codes. Proc IEEE DSP2000, Hunt, Tx, USA, 2000.

[113] http://www.inference.phy.cam.ac.uk/mackay/codes/data.html. *On Line*.

[114] www.altera.com. Cyclone II FPGAs. *On Line*.

[115] M.K. Ku; H.S. Li; Y.H. Chien. Code design and decoder implementation of low density parity check code. *Emerging Information Technology Conference*, 2005. ISBN 0780393287.

[116] www.altera.com. Quartus II Software. *On Line*.

[117] P. Vontobel; R. Koetter. On low-complexity linear-programming decoding. *European Transactions on Telecommunications*, 18:509–517, 2007.

[118] L. Arnone; C. Gayoso; C. González; J. Castiñeira Moreira; L. Liberatori. Analisys and FPGA implementation of combined error-control and encryption schemes. *SPL07, III Southern Conference on Programmable Logic. Mar del Plata, Argentina*, pages 87–90, 2007.

[119] L. Arnone; C. Gayoso; C. González; J. Castiñeira Moreira; L. Liberatori. Uso de códigos de decodificación iterativa para mejorar la transmisión de mensajes cifrados con AES. *XIII Workshop de IBERCHIP. Lima. Perú*, pages 245–248, 2007.

[120] L. Arnone; C. Gayoso; C. González; J. Castiñeira Moreira; M. Liberatori. Security and BER performance trade-off in wireless communication systems applications. *XII RPIC, XII Reunión de Trabajo en Procesamiento de la Información y Control. Río Gallegos, Argentina*, page 56, 2007.

[121] J. Edney; W. A. Arbaugh. Real 802.11 security, WI-FI protected access and 802.11i. Addison Wesley, 2004.

[122] B. Brown. 802.11. *IEEE Potentials*, pages 25–27, 2003.

[123] J. Castiñeira Moreira; D. Petruzzi; M. Liberatori; B. Honary. Trellis hopping turbo coding. *Communications, IEE Proceedings*, 153:966–975, 2006.

[124] J.C. Bonadero; M. Liberatori;H. Villagarcía Wanza. Expansión de la clave en rijndael. diseño y optimización en VHDL. *Anales XI RPIC, XI Reunión de Trabajo en Procesamiento de la Información y Control*, pages 115–120, 2005.

[125] M. Liberatori; J.C. Bonadero. Minimum area, low cost fpga implementation of AES. *VIII International Symposium on Communications Theory and Applications, UK*, pages 461–466, 2005.

[126] M. Liberatori; J.C. Bonadero. AES 128 cipher. minimum area, low cost FPGA implementation. *SPL 06, II Southern Conference on Programmable Logic*, pages 51–58, 2006.

[127] P. Mroczkowski. Implementation of the block cipher rijndael using altera FPGA. http://csrc.nist.gov/encryption/aes/round2/comments/20000510-pmroczkowski.pdf, 2001.

[128] G. Masera; G. Piccinini; M. Ruo Roch; M. Zamboni. VLSI architectures for turbo codes. *IEEE Tran. on Very Large Scale Integration VLSI Systems*, 7:369–379, 1999.

[129] S. Hong; J. Yi; W.E. Stark. VLSI design and implementation of low complexity adaptive turbo code encoder and decoder. *IEEE Workshop on Signal Processing Systems*, pages 233–242, 1998.

[130] S. Hong; W.E. Stark. VLSI circuit complexity and decoding performance analysis for low - power RSC turbo code and iterative block decoders design. *Proc. of the IEEE Military communications Conf.*, 3:708–712, 1998.

[131] www.actel.com. *On Line.*

[132] www.quicklogic.com. *On Line.*

[133] C. M. González. *Diseño de Bloques Caóticos Orientados a Dispositivos Lógicos Programables.* PhD thesis, Universidad Nacional de Mar del Plata, Mar del Plata, Argentina, Junio 2006.