Comunicaciones por Satélite
Curso 2008-09

Codificación de canal

Codificación por bloques

Miguel Calvo Ramón

Ramón Martínez Rodríguez-Osorio

Índice

- **Introducción. Concepto y objetivo de la codificación de canal**
- **Sistemas basados en ARQ**
- **Modelos de canal. Decisión dura vs. Decisión blanda**
- **Capacidad del canal. Límite de Shannon**
- **Esquemas FEC**
  - Parámetros característicos de un código
  - Códigos de bloques. Códigos de Hamming, Golay, BCH, Reed-Solomon
  - Códigos convolucionales. Decodificador de Viterbi. Ejemplo
  - Aspectos de implementación
- **Modulación codificada (TCM)**
- **Turbocódigos**
- **Códigos LDPC**

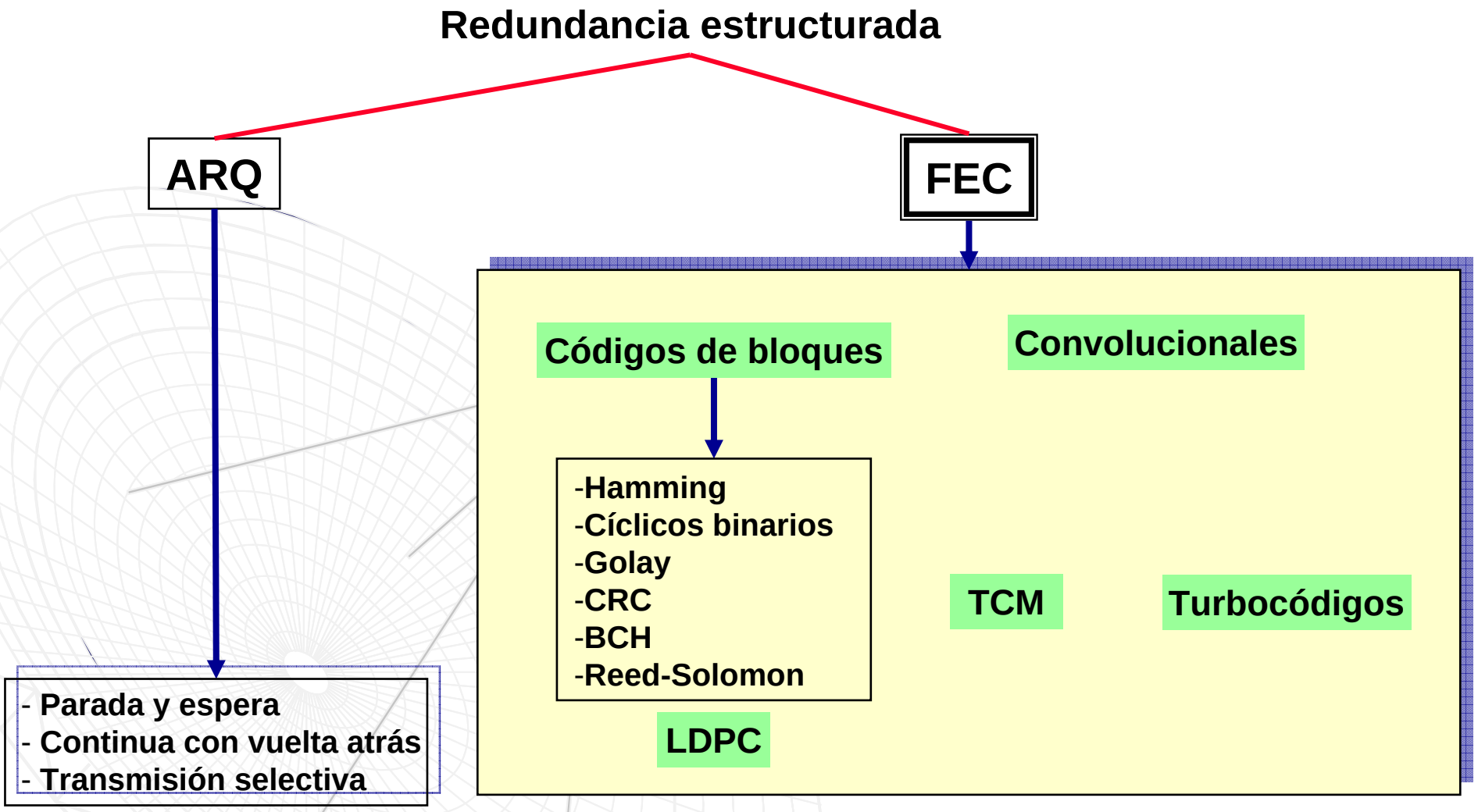
Codificación

- **En sistemas digitales puede mejorarse la BER usando técnicas de corrección de errores. En sistemas analógicos estas técnicas no existen o requieren tiempos de proceso muy grandes.**

- **La *redundancia estructurada* es un método de insertar símbolos adicionales a los de información del mensaje. La unicidad de la redundancia estructurada permite tolerar el que varios símbolos de información puedan ser erróneos sin destruir la unicidad de la información que contienen, lo que causaría un bloque de error.**

- **El *promediado de ruido* se obtiene haciendo depender los símbolos redundantes de un conjunto de varios símbolos de información.**

Familias de códigos

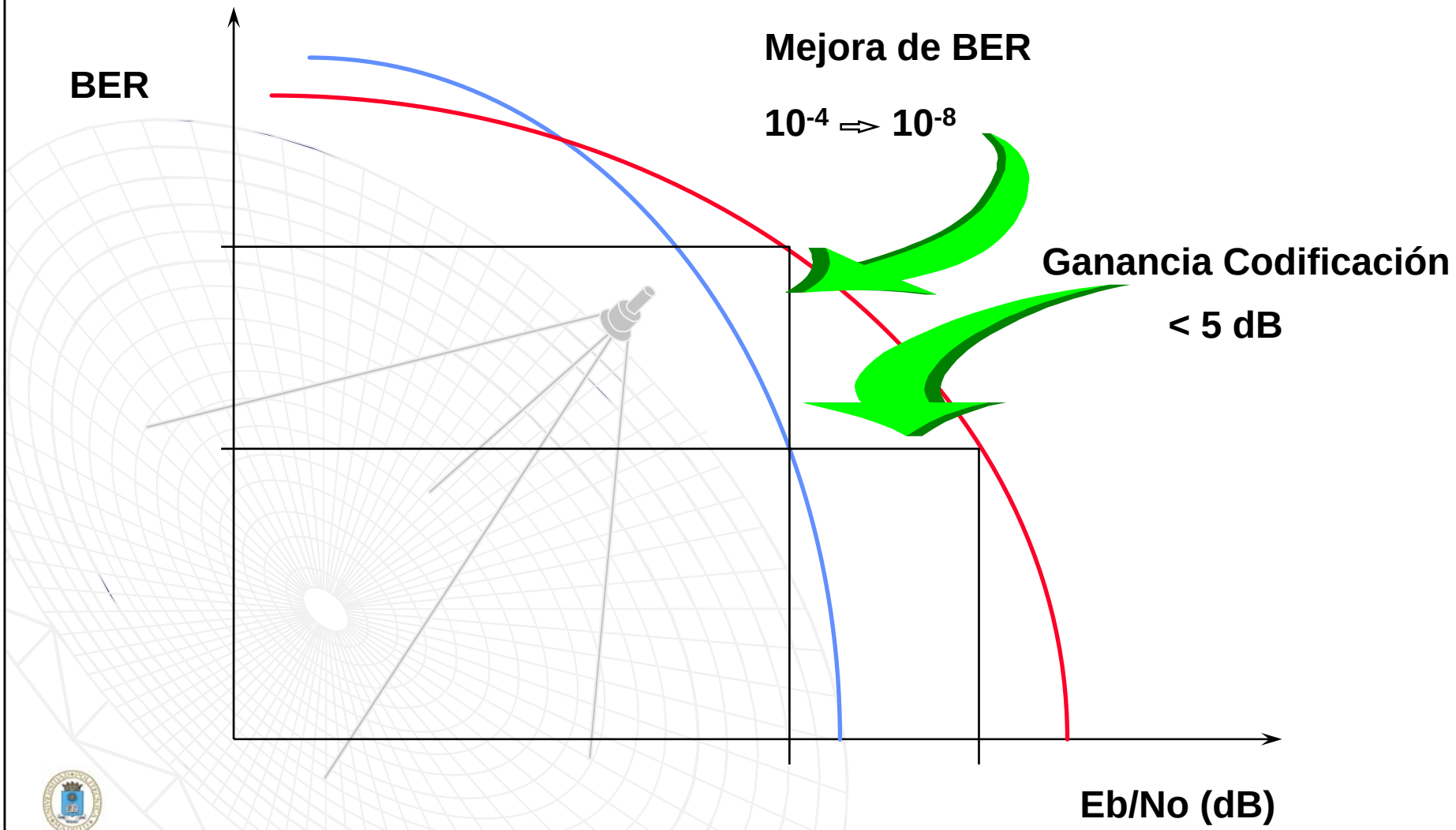


Usos de la Codificación

- ***Detección de error y retransmisión*** **(ARQ)**
  - Utiliza los bits de paridad (bits redundantes añadidos a los de datos) para detectar la presencia de errores.
  - El terminal receptor no intenta corregir los errores y simplemente pide al transmisor una retransmisión de los datos.
  - Se requiere por tanto un enlace doble (ida y vuelta) para establecer este diálogo entre transmisor y receptor.

- ***Corrección adelantada de errores*** **(FEC)**
  - Necesita sólo un enlace de ida
  - Los bits de paridad se diseñan tanto para detectar la presencia de errores como para corregirlos.
  - No es posible corregir todos los posibles errores.

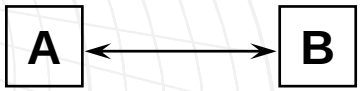
Ganancia de Codificación



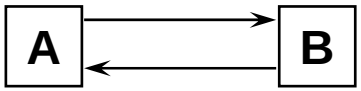
Tipos de Conexión

- ***Simplex***
  - Un terminal A se conecta a otro B en una sola dirección
  - La transmisión se realiza siempre de A a B y nunca en sentido inverso.
- ***Semiduplex***
  - Las transmisiones pueden realizarse entre los dos terminales en ambas direcciones pero no simultáneamente.
- ***Duplex***
  - Hay un doble enlace entre los dos terminales
  - Las transmisiones pueden realizarse en las dos direcciones simultáneamente.

Simplex



Semiduplex



Duplex

Petición Automática de Retransmisión (ARQ)

- ***ARQ de parada y espera***
  - El transmisor envía un bloque de información y espera del receptor una señal de reconocimiento (ACK), que le indica que la información se ha recibido sin errores, antes de que proceda al envío de más información.
  - Si se detectan errores, el receptor envía una señal de no reconocimiento (NAK) y el transmisor retransmite la información.
  - Sólo requiere una conexión semiduplex.
- ***ARQ continua con vuelta atrás***
  - El transmisor envía continuamente paquetes de información, con un número de identificación, y el receptor envía los correspondientes ACK o NAK con la identificación del paquete.
  - Cuando se recibe un NAK el transmisor vuelve a retransmitir desde el bloque erróneo en adelante.
  - Se requiere una conexión dúplex.
- ***ARQ con repetición selectiva***
  - Parecido al anterior con la diferencia de que sólo se retransmiten los mensajes recibidos con error.

ARQ versus FEC (1)

- **Con errores a ráfagas: ARQ funciona mejor que los FEC simples**
- **Con errores independientes: FEC funciona mejor que ARQ**
- **Si se conoce la naturaleza de los errores, puede diseñarse un sistema FEC ad-hoc más eficiente que ARQ**
- **FEC es muy sensible a la degradación del canal (interferencia, ruido impulsivo, atenuación, etc.)**
  - Las curvas de BER vs. Eb/No tienen mucha pendiente
- **ARQ requiere grandes buffers de memoria e introduce retardo, mientras que en los sistemas FEC el throughput se mantiene constante**

ARQ versus FEC (2)

- **La ventaja del ARQ sobre el FEC es que el equipo de detección de errores es mucho más simple y se requiere el uso de menos redundancia en los códigos.**

- **<u>Debe</u> usarse FEC cuando:**
  - la conexión es simplex,
  - siendo semiduplex los retardos con ARQ sean excesivos,
  - el número esperado de errores (sin corrección) implique un número excesivo de retransmisiones.

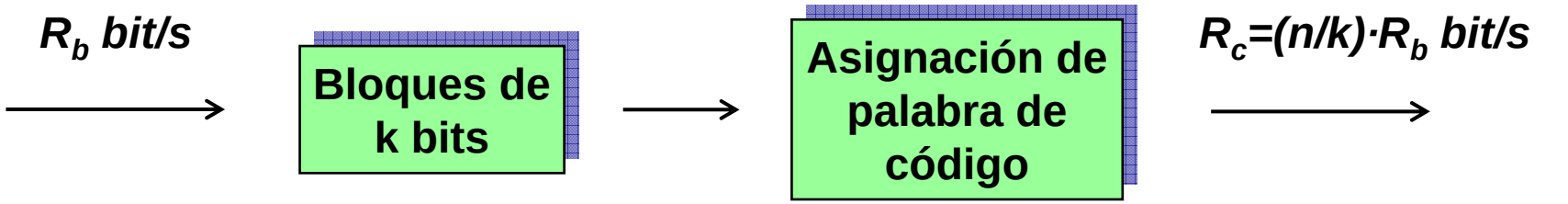
Códigos de Bloques

- **Los datos (símbolos binarios o bits) se segmentan en bloques de k-bits, donde k es la longitud de bloque.**
- **Cada bloque de información puede representar uno de $M = 2^k$ mensajes diferentes.**
- **El codificador transforma cada bloque de información en un bloque mayor de n bits (n > k) añadiendo n-k bits redundantes de una manera predeterminada.**
- **Cada bloque de n bits constituye *una palabra de código* contenida en el conjunto de M posibles valores.**
- **Las palabras de código se modulan y transmiten al canal.**
- **Un codificador de bloques es un dispositivo sin memoria en el sentido de que cada n bits de código dependen solamente de un bloque de k bits de información específico y no de otros.**

Códigos de Bloques

- La relación entre los bits de información y los bits totales de una palabra de código se denomina *tasa de codificación* R = k/n.
- Si R_b es la tasa de bits de información a la entrada la tasa de bits codificados a la salida es: $R_c = nR_b/k$.
- n varía entre 3 y algunos cientos; R entre 1/4 y 7/8.
- La corrección de errores puede hacerse con códigos de bloques (n,k) porque sólo pueden transmitirse $M = 2^k$ posibles palabras de código y el número de posibles palabras recibidas 2^n es mucho mayor.
- La unicidad con que se añaden los n-k bits redundantes permite al decodificador identificar el mensaje que se ha transmitido.

Códigos de Bloques



10101010101101010…

k k k k … k

(k=3)

11100011011100110011…

n n … n

(n=5)

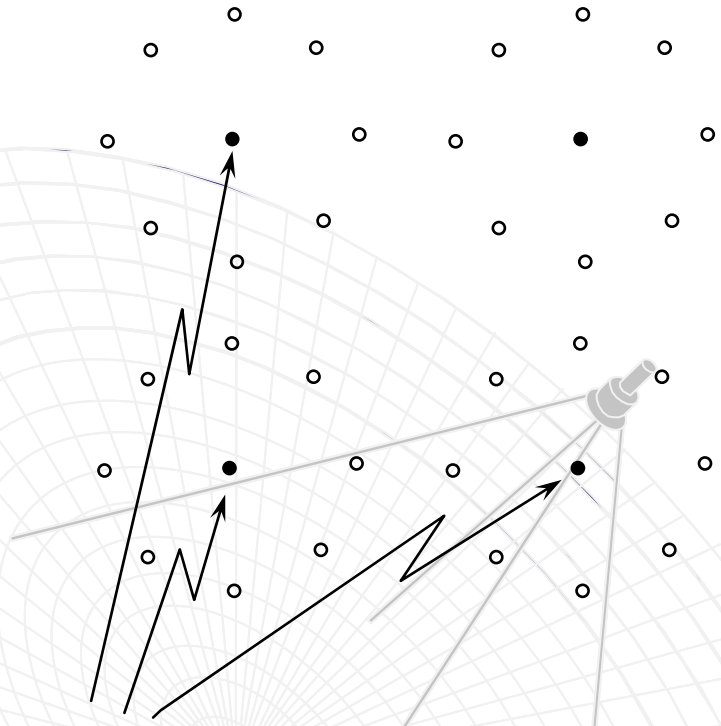
Parámetros del código:

- **k=longitud de los bloques**
- **n=longitud de la palabra código**
- **(n-k)=número de bits de redundancia**
- **R=k/n=tasa de codificación**
- **R_c: velocidad binaria a la salida del codificador**

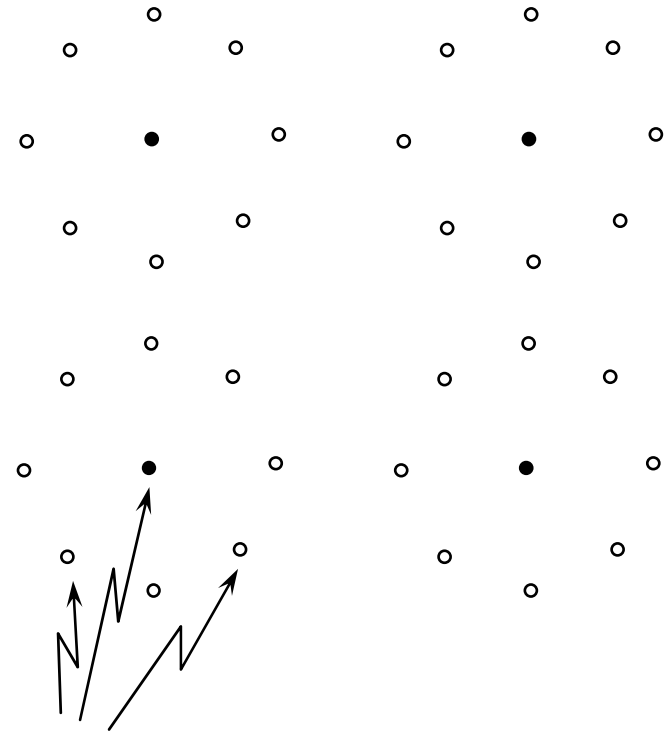
Dispositivos sin memoria (la palabra código sólo depende del bloque actual).

- **Hay una correspondencia biunívoca entre los bloques de información y las palabras código.**

Redundancia de la Codificación



$M=2^k$ Palabras de Código

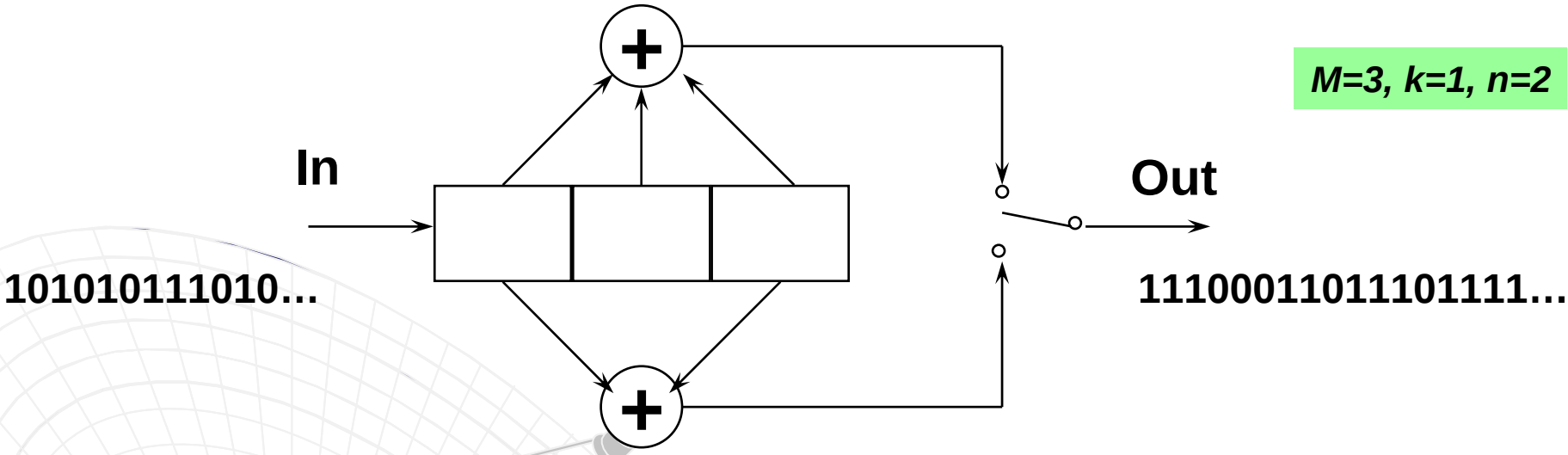


2^n Mensajes Recibidos

Codificación Convolucional

- **Los datos de información se hacen pasar por un registro de desplazamiento de M etapas que desplazan k bits al tiempo.**
- **Para cada M bits de información almacenados en el registro de desplazamiento hay n circuitos lógicos que operan sobre el contenido del registro de desplazamiento para producir n bits de código como salida.**
- **La tasa de codificación es por tanto R = k/n.**
- **Un bit de información permanece en el registro de desplazamiento durante M/k cambios e influye en el valor de nM/k bits de código. Por ello, el codificador convolucional es un dispositivo con memoria.**
- **Valores típicos de k y n están entre 1 y 8, para R entre 1/4 y 7/8 y para M entre 5 y 70.**

Codificación convolucional

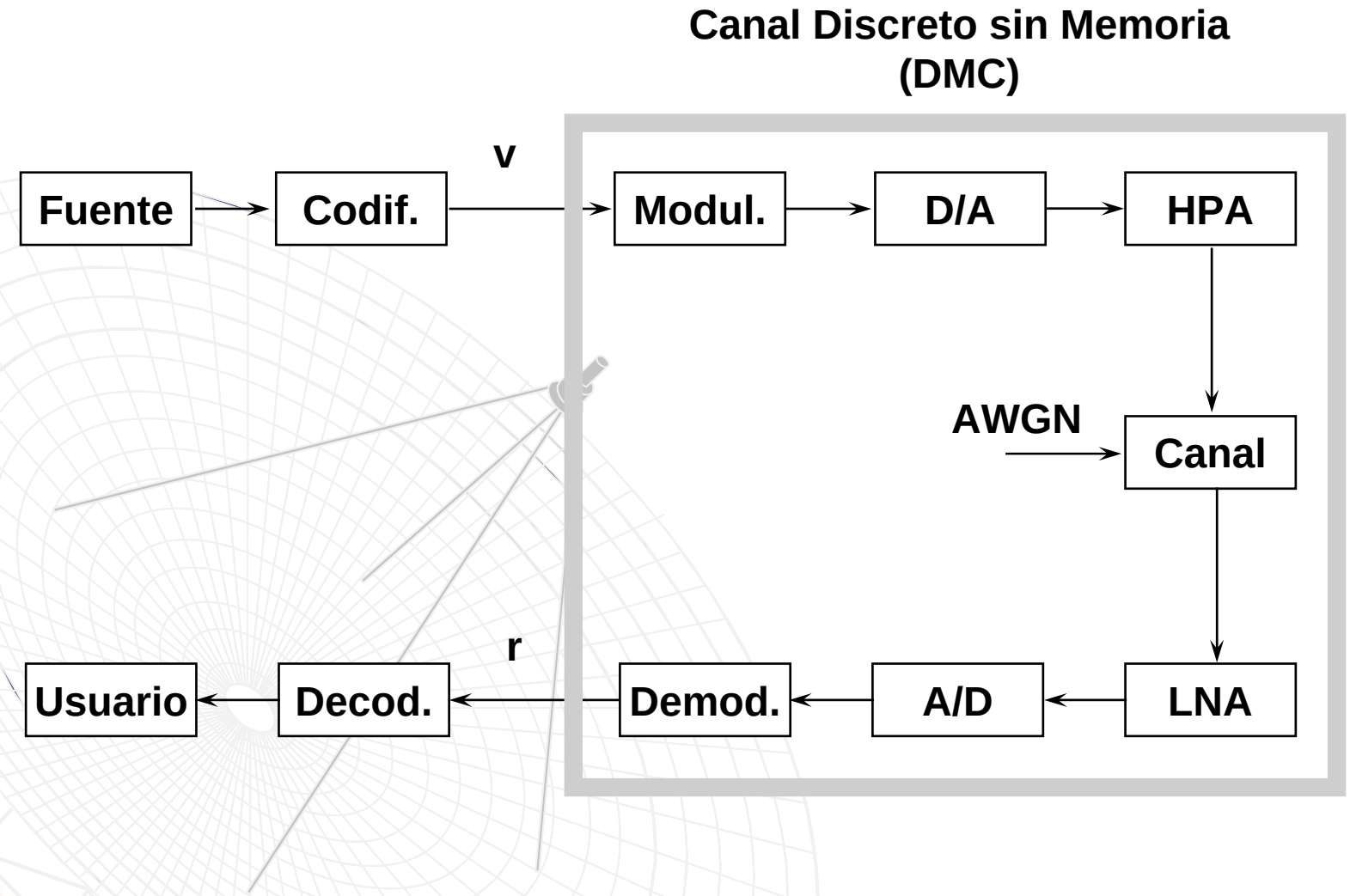


Parámetros del código:

- **M=número de etapas del registro de desplazamiento**
- **k=número de símbolos de entrada**
- **n=número de circuitos lógicos**
- **R=k/n=tasa de codificación**
- **R_c: velocidad binaria a la salida del codificador**

Un bit de información permanece en el registro de desplazamiento durante M/k cambios e influye en el valor de nM/k bits de código. Por ello, el codificador convolucional es un dispositivo con memoria.

Canal Discreto



Canal Discreto

El canal discreto sin memoria (DMC) se caracteriza por:

un conjunto de símbolos de entrada M-arios, que son invariantes e independientes símbolo a símbolo.

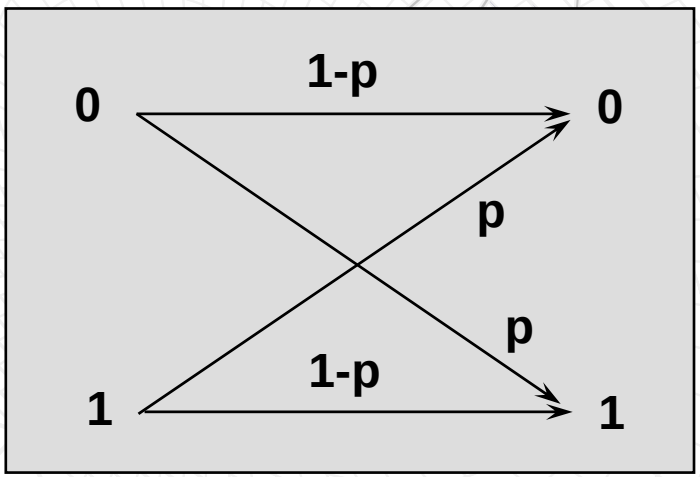
símbolos de salida Q-arios

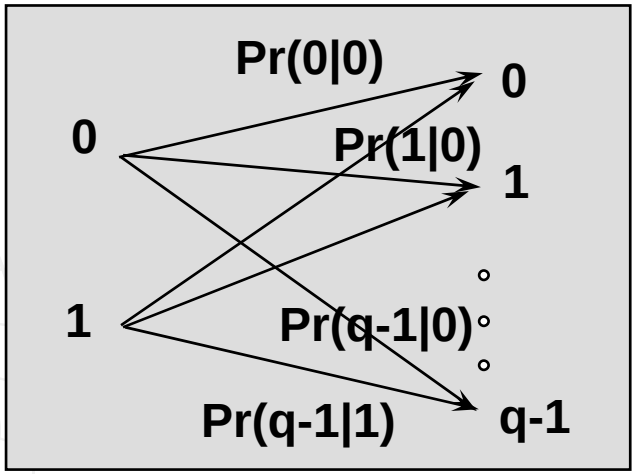
probabilidades de transición Pr(j|i), 0<= i <= M-1, 0 <=j <= Q-1

donde i es un símbolo a la entrada del modulador,

j es un símbolo a la salida del demodulador y

Pr(j|i) es la probabilidad de recibir j cuando se ha transmitido i





Canal Binario Simétrico con Decisión Dura

- **El canal DMC más usual es canal binario simétrico BSC**
  - se utiliza modulación binaria (M=2),
  - la salida del demodulador está cuantificada a Q=2 niveles
  - Pr(0|1) = Pr(1|0) = p y Pr(1|1) = Pr(0|0) = 1 - p.

- **La probabilidad de transición p puede calcularse a partir de la forma de onda usada en la transmisión, la función densidad de probabilidad de ruido y el umbral de cuantificación de salida del demodulador.**

- **Por ejemplo, cuando se utiliza BPSK con demodulación coherente la probabilidad de transición es el valor medio de la probabilidad de error de bit.**

- **El umbral óptimo es 0 y la salida es 0 si la tensión de salida del filtro adaptado es negativa.**

- **El proceso de decodificación se llama *decodificación con decisión dura*.**

Canal Binario Simétrico con Decisión Blanda

- **Si la salida del demodulador utiliza cuantización con Q >2, o incluso se deja sin cuantizar, se dice que el demodulador toma *decisiones blandas*.**
- **El decodificador acepta señales multinivel (o analógicas) y el proceso de decodificación se denomina *decodificación blanda*.**

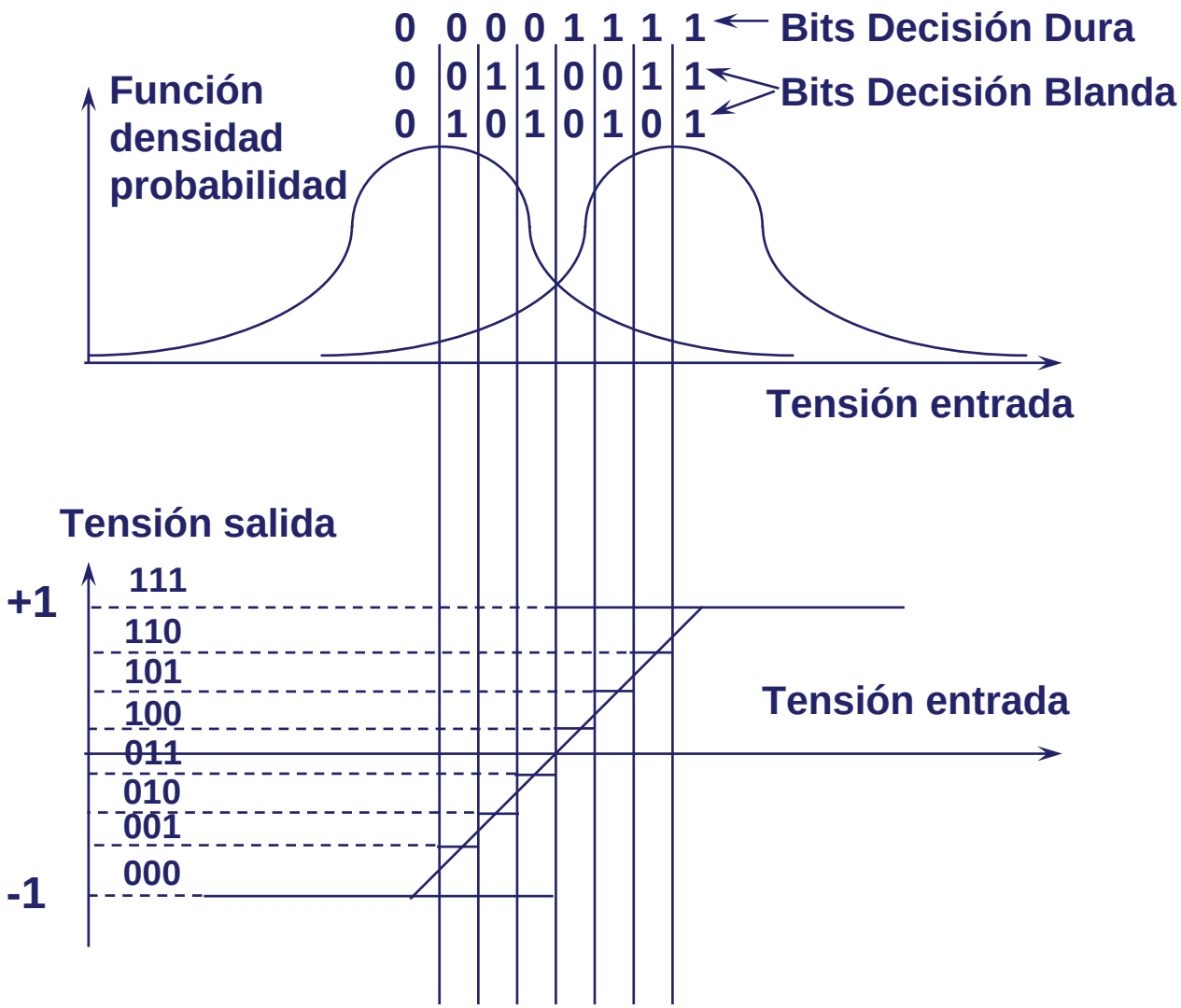
- **Un decodificador de decisión blanda es más complejo que uno de decisión dura**
  - Se necesita un control automático de ganancia y además hay que manipular $\log_2 Q$ bits por cada bit del canal.

- **La decisión blanda ofrece ganancia de codificación adicional de 2 dB sobre la decisión dura, para valores realistas de Eb/No.**

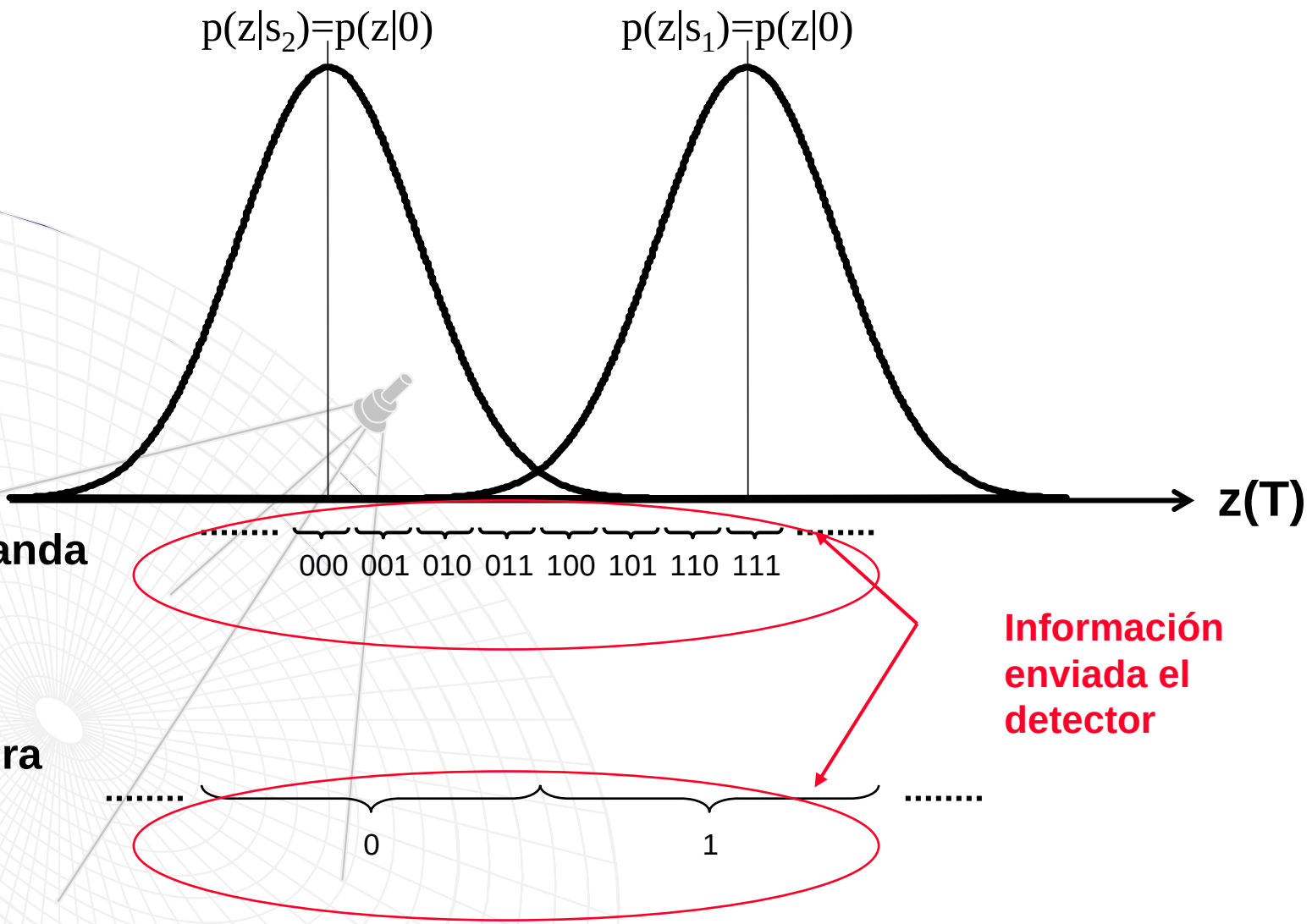
- **En la práctica se usan normalmente 8 niveles de cuantización ya que al tomar más niveles se obtiene poca ganancia adicional (~0.2 dB).**

- **Las salidas cuantizadas a ocho niveles tienen un umbral de decisión y tres pares de umbrales de confianza.**

Decisión Dura vs Blanda



Decisión Dura vs Blanda



Capacidad del Canal

Para un canal con ruido aditivo blanco gaussiano la capacidad, según el teorema de Shannon, es:

$$H = B\log_2\left(1+\frac{C}{N_0 B}\right) \quad bps$$

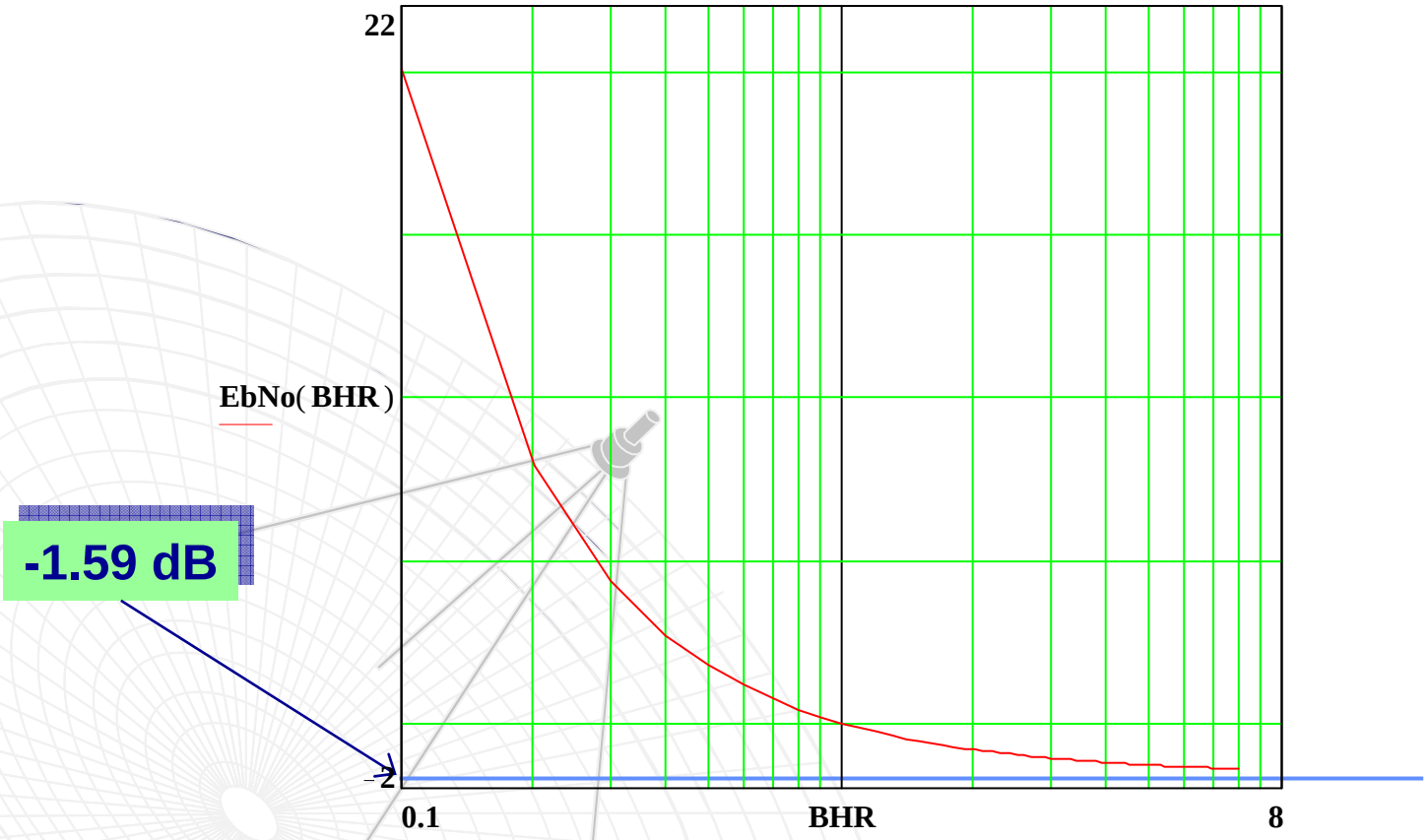
siendo B= el ancho de banda del canal en Hz.
C= la potencia de la señal en watios.
No = la densidad espectral de potencia de ruido en banda única.

Para un canal digital (binario): H=1/Tb y Eb = C×Tb = C/H, siendo Tb la duración de bit. Por tanto:

$$\frac{H}{B} = \log_2\left(1+\frac{E_b}{N_0}\times\frac{H}{B}\right) \Rightarrow \frac{E_b}{N_0} = \frac{2^{\frac{H}{B}}-1}{\frac{H}{B}}$$

La relación H/B es la *eficiencia espectral* del enlace/canal.

Límite de Shannon



El límite H/B → 0 (B/H → ∞) es Eb/No = ln 2, y en dB: 10log(ln 2) = -1.59 dB. Por tanto, independientemente del ancho de banda utilizado, no se puede transmitir información libre de errores por debajo del límite Eb/No = -1.59 dB.

Límite de Shannon vs QPSK

En la práctica se trabaja lejos del límite teórico. Por ejemplo, una transmisión QPSK con una BER de 10^{-10} utiliza una eficiencia espectral H/B de 2 y una Eb/No que puede obtenerse de la expresión:

$$BER = \frac{1}{2}\,erfc\left(\sqrt{\frac{E_b}{N_0}}\right) \cong \frac{1}{2}\frac{1}{\sqrt{\pi}}\exp\left(-\frac{E_b}{N_0}\right) = 0.282\exp\left(-\frac{E_b}{N_0}\right)$$

$$\text{BER} = 10^{-10} : \frac{\text{Eb}}{\text{No}} = \ln\left(\frac{0.282}{10^{-10}}\right) = 21.76 = 13.38\ \text{dB}$$

$$\text{Límite de Shannon} : \frac{E_b}{N_0} = \frac{2^{\frac{H}{B}} - 1}{\frac{H}{B}} = \frac{2^2 - 1}{2} = 1.5 = 1.76\ \text{dB}$$

Margen 11.62 dB

La codificación permite, sobre todo para condiciones Eb/No bajas, mantener la BER sin incrementar excesivamente el ancho de banda, acercándonos al límite teórico.

Codificación para detección de errores

- **Es una técnica por la que añadiendo bits redundantes a una secuencia de datos podemos detectar un error en la misma.**
- **Salvo que se añada un bit redundante por cada bit de datos no es posible el detectar con esta técnica cuál es el bit erróneo.**
- **Normalmente se añade un bit redundante por cada N bits de datos con lo que se puede detectar la presencia de un error en el bloque de N bits.**
- **Como ejemplo, consideraremos el conocido código ASCII de 8 bits, con uno de paridad. El código contiene 128 (2^7) caracteres, cada uno de 7 bits de datos más uno de paridad.**
- **Los 7 bits de datos permiten representar el alfabeto en mayúsculas y minúsculas, los números del 0 al 9, los signos de puntuación y adicionalmente algunos comandos.**

Control de Paridad

- La paridad puede ser par, en cuyo caso la suma de todos los bits es par (o cero si la suma se hace en módulo 2), o impar en cuyo caso la suma de todos los bits debe ser 1.
- En un sistema con paridad par si la suma de todos los bits de un carácter es par entonces éste se ha recibido sin error, o bien se ha recibido con 2 o más (número par) errores, en cuyo caso no somos capaces de detectarlos.
- Si la tasa de bit erróneo en el enlace es p ($p < 10^{-1}$), la probabilidad de que k bits de un bloque de n sean erróneos $P_e(k)$ es:

$$P_e(k) = \binom{n}{k} p^k (1-p)^{n-k}$$

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Probabilidad de Error

- **En el caso de un código ASCII la probabilidad de que haya 2 bits erróneos será mucho mayor que la de que haya 4 y por tanto la probabilidad de que haya un error no detectado será:**

$$P_{wc} \approx P_e(2) = \binom{8}{2} p^2 (1-p)^6 \approx 28p^2$$

si la probabilidad de bit erróneo en el enlace es p << 1.

- **Si no se hubiese utilizado el bit de paridad la probabilidad de tener un error en el carácter de 7 bits sería:**

$$P_{uc} \approx \binom{7}{1} p_u (1-p_u)^7 \approx 7p_u$$

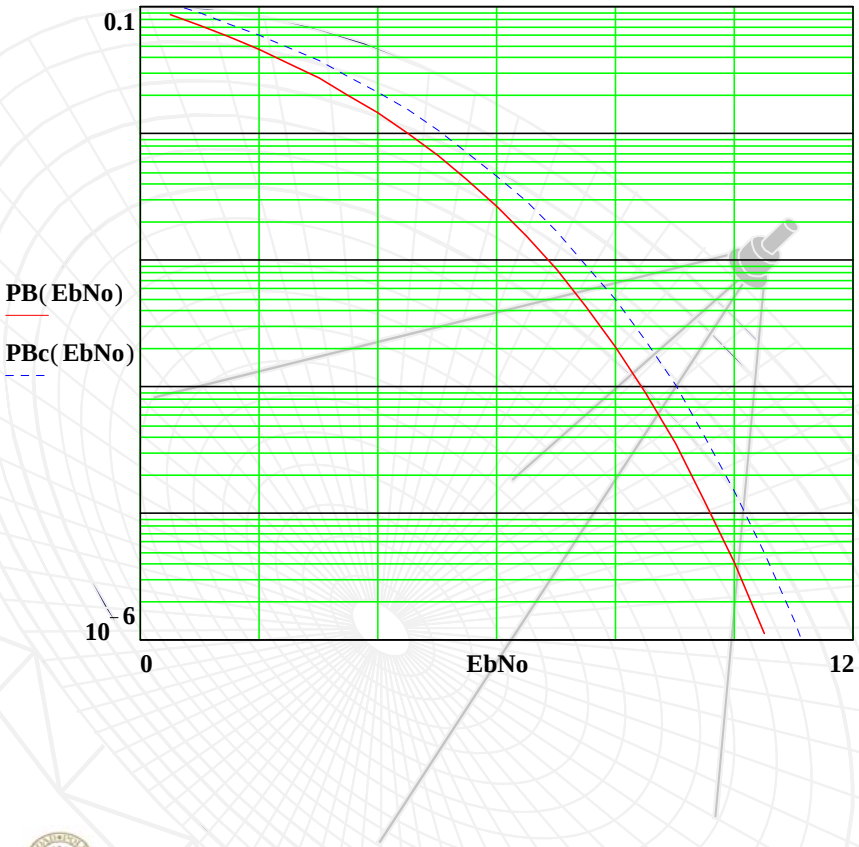
- **Por tanto, si $p \approx p_u$, la mejora de codificación sería:**

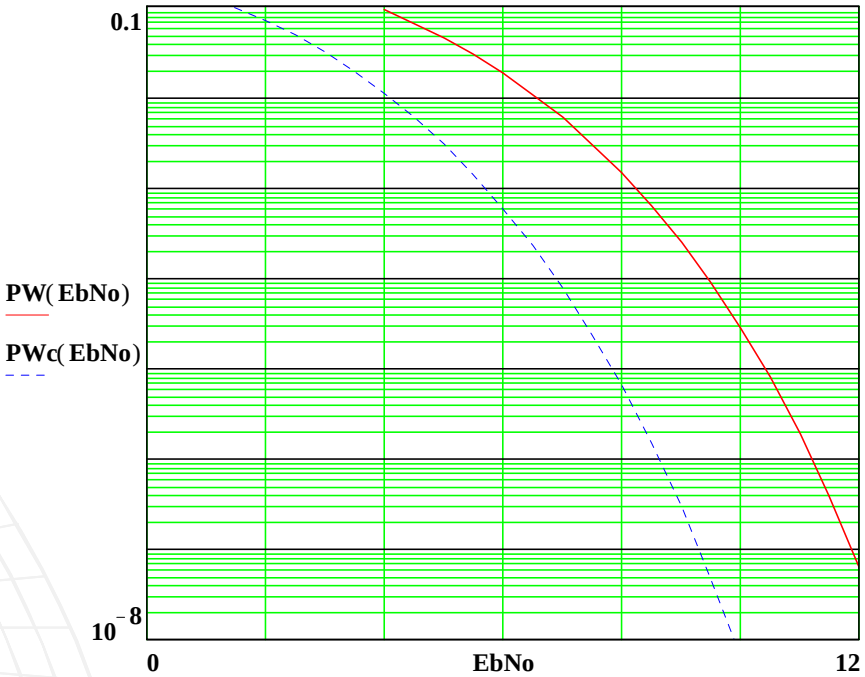
$$\frac{28p^2}{7p_u} \approx 4p$$

Probabilidad de Error

- **Si por ejemplo p = 10^{-3} , la probabilidad de error sin codificar sería $7x10^{-3}$.**
- **Mientras que con control de paridad sería $28 \times 10^{-3} \times 10^{-3} = 2.8 \times 10^{-5}$.**
- **Si fuese p = 10^{-6} entonces la probabilidad de error no detectado sería de $2.8x10^{-11}$, que es muy pequeña.**

BER en el canal con y sin Codificación

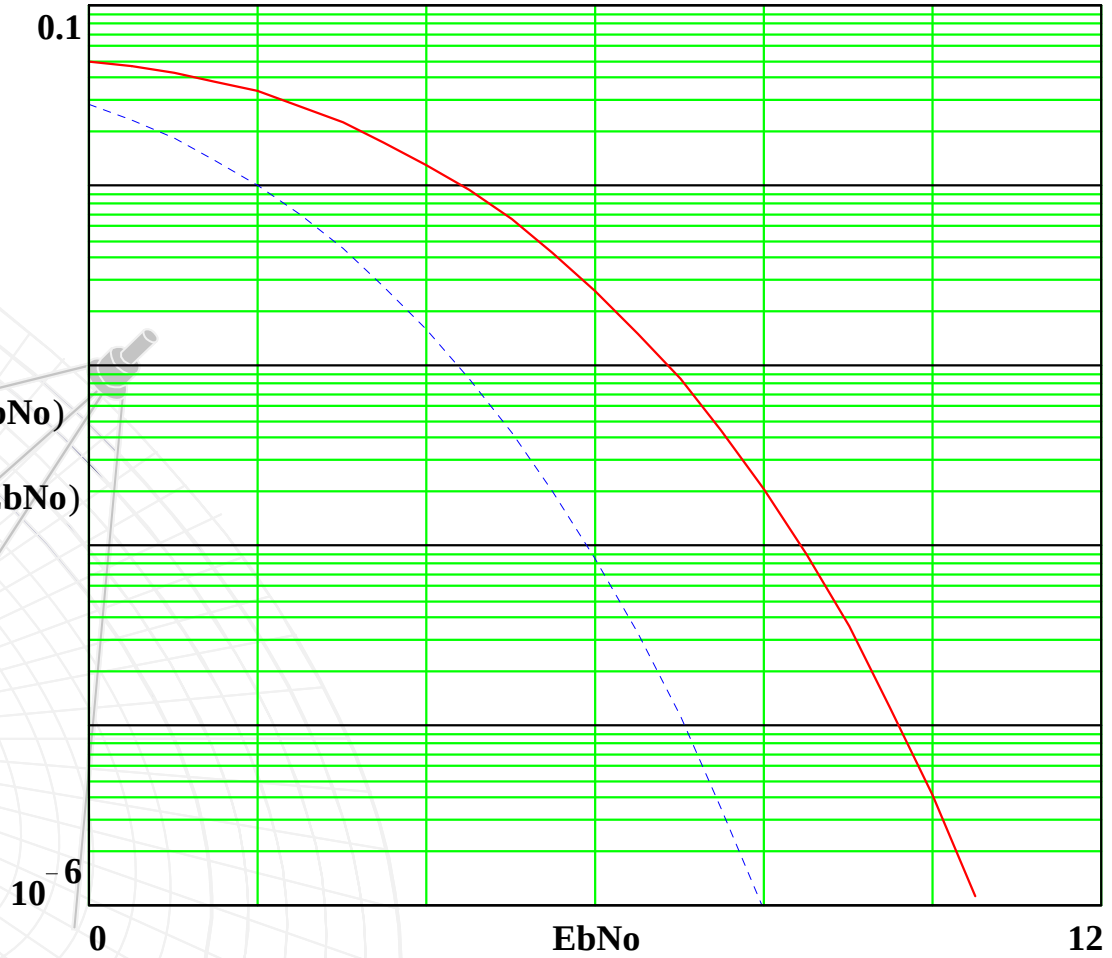




Ganancia de Codificación

$$P_b = \frac{P_w}{7}$$

$$P_bc = \frac{P_{wc}}{7}$$



Códigos de Bloques (n,k)

Un código de bloques es aquél en el que se pueden enviar 2^k mensajes distintos, de k bits cada uno, a los que se añaden (n-k) bits redundantes que se generan a partir de los k bits del mensaje siguiendo una regla predeterminada.

Un código de bloques es *sistemático* si los primeros (últimos) k bits son el mensaje y los restantes (n-k) son los de chequeo. Al bloque de n bits se le llama palabra de código. Los códigos de bloques se designan como (n,k), p.e. el código ASCII sería un código (8,7).

Un código de bloques es *lineal* cuando la suma módulo 2 de dos palabras de código es también otra palabra de código. La forma general de obtener una palabra de código lineal es:

$$C = D(1,k) \times G(k,n)$$

donde *D es un mensaje* (vector de k elementos) y *G es la matriz generadora* (con k filas y n columnas). Al multiplicar ambas (módulo 2) se obtiene la palabra de código de n elementos.

La forma general de G, para un código sistemático, es:

$$G = \left[I_k \middle| P_{k,n-k} \right]_{k \times n} \qquad G = \left[P_{k,n-k} \middle| I_k \right]_{k \times n}$$

donde I_k *es la matriz identidad*. De esta forma, al multiplicar por D, los k primeros (últimos) elementos de C son los de D.

Chequeo de paridad y Síndrome

Las palabras codificadas se llaman también vectores de código.

Para detectar errores en los vectores de código usamos la *matriz de chequeo de paridad H:*

$$\mathbf{H} = \left[\mathbf{P}^T \middle| \mathbf{I}_{n-k}\right]_{(n-k)\times n} \qquad \mathbf{H} = \left[\mathbf{I}_{n-k} \middle| \mathbf{P}^T\right]_{(n-k)\times n}$$

siendo P^T la traspuesta de $P_{k,n-k}$ e I_{n-k} la matriz identidad.

La detección del error se obtiene multiplicando la palabra de código recibida R por la traspuesta de la matriz de chequeo de paridad RxH^T.

Dado que todas las palabras de código C son tales que: $CxH^T = 0$, si el resultado de RxH^T es cero la palabra recibida será una palabra del código. Si el producto no es cero habrá un error E de manera que R=C+E.

El producto RxH^T recibe el nombre de *síndrome S*. El síndrome es igual a:

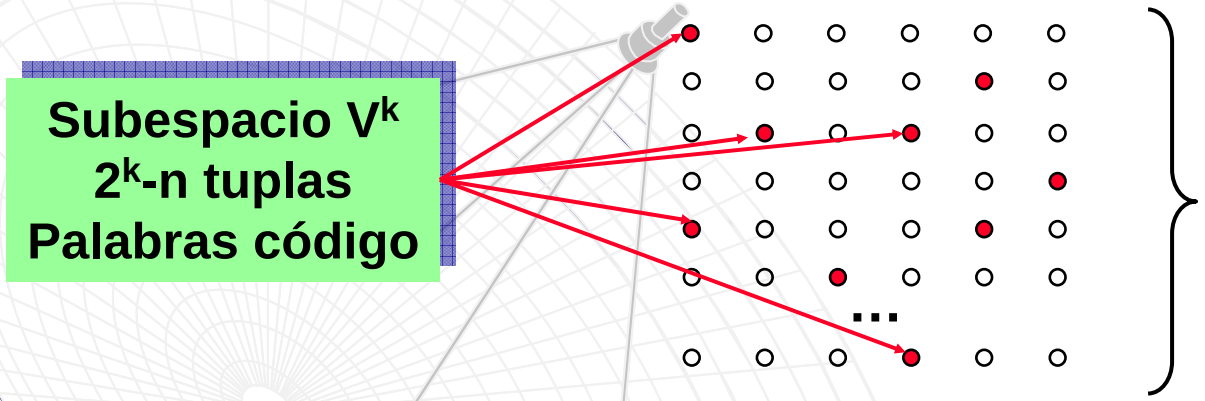
$$\boldsymbol{S = R \times H^T = (C + E) \times H^T = C \times H^T + E \times H^T}$$

y por tanto será cero si no hay error (E=0). Por contra, si no es cero, habrá error.

Códigos de Bloques

Mensaje → bloques de k bits → 2^k mensajes distintos
Palabra código → bloques de n bits (n>k) → 2^k palabras código distintas

El conjunto de palabras código deben ser un subespacio de V^n:
(a) Debe contener el elemento 0 0 0 0 … 0 0
(b) Operaciones de suma y producto en modulo-2
(c) La suma de dos palabras código debe pertenecer al subespacio (lineal)



Ejemplo:
Espacio V^4={0000 0001 0010 0011 0100 0101 0110 0111
1000 1001 1010 1011 1100 1101 1110 1111}

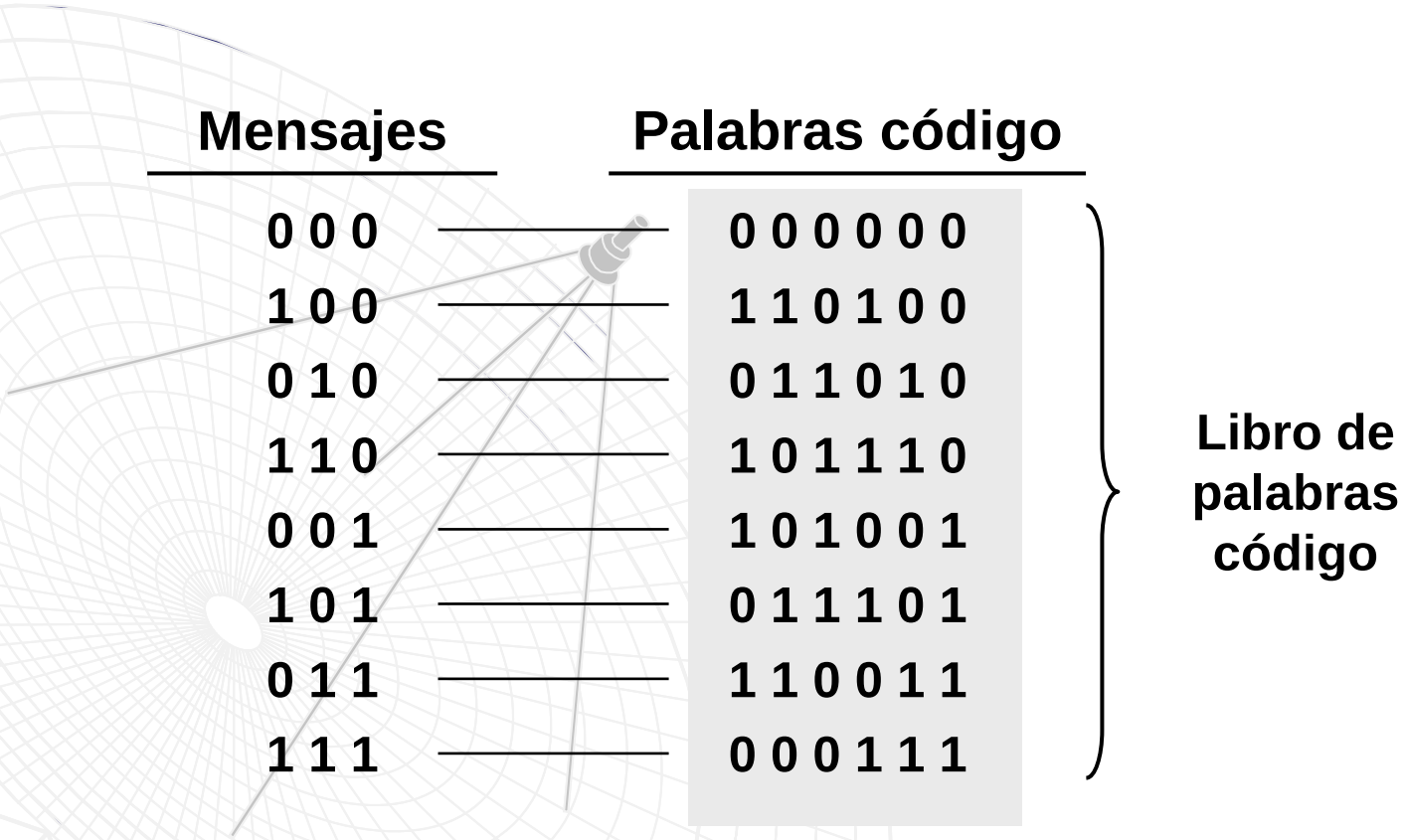
Subespacio S4={0000 0101 1010 1111}

Códigos de Bloques. Ejemplo

Ejemplo de código de bloques lineal: (n,k)=(6,3)

2^k=8 palabras código

Espacio V^6=64 6-tuplas



Matriz generadora G

Dos opciones de <u>implementación</u>:

1) Look-up tables → 2^k entradas

Si (127,92) → 2^{92}~$5\cdot10^{27}$ palabras código ⇒ Memoria, Tiempo de búsqueda

2) Matriz generadora del código G

No requiere almacenamiento masivo de palabras código

Las 2^k codewords son un subespacio → Puede encontrarse una <u>base</u> de k vectores <u>linealmente independientes</u> (V_1, V_2, …, V_k)

Cualquier palabra código U se obtiene como (m_i=0, 1):

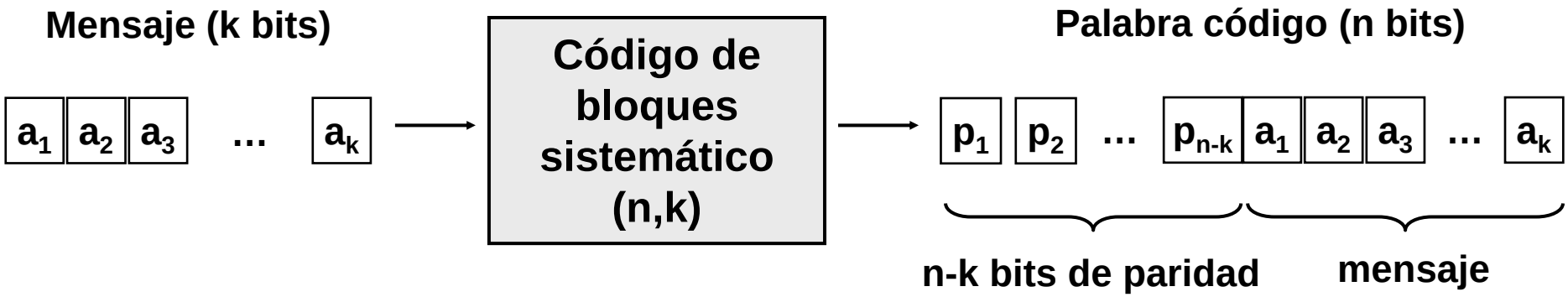
$$\mathbf{U} = \mathbf{m}_1\mathbf{V}_1 + \mathbf{m}_2\mathbf{V}_2 + \ldots + \mathbf{m}_k\mathbf{V}_k$$

La matriz generadora de código (k×n) se define como:

$$\mathbf{G} = \begin{bmatrix} \mathbf{V}_1 \\ \mathbf{V}_2 \\ \cdots \\ \mathbf{V}_k \end{bmatrix} = \begin{bmatrix} \mathbf{v}_{11} & \mathbf{v}_{12} & \cdots & \mathbf{v}_{1n} \\ \mathbf{v}_{21} & \mathbf{v}_{22} & \cdots & \mathbf{v}_{2n} \\ \cdots & & & \\ \mathbf{v}_{k1} & \mathbf{v}_{k2} & \cdots & \mathbf{v}_{kn} \end{bmatrix}_{k\times n} \longrightarrow \mathbf{U} = \mathbf{mG}$$

Códigos bloque sistemáticos

Un código de bloques es *sistemático* si los primeros (últimos) k bits son el mensaje y los restantes (n-k) son los de chequeo.



Mensajes	Palabras código
0 0 0	0 0 0 0 0 0
1 0 0	1 1 0 1 0 0
0 1 0	0 1 1 0 1 0
1 1 0	1 0 1 1 1 0
0 0 1	1 0 1 0 0 1
1 0 1	0 1 1 1 0 1
0 1 1	1 1 0 0 1 1
1 1 1	0 0 0 1 1 1

La forma general de G, para un código sistemático, es:

$$\mathbf{G} = \left[\mathbf{I}_k \middle| \mathbf{P}_{k,n-k}\right]_{k\times n}$$

$$\mathbf{G} = \left[\mathbf{P}_{k,n-k} \middle| \mathbf{I}_k\right]_{k\times n}$$

Matriz G para códigos sistemáticos

En el ejemplo anterior de código (6,3):

$$\mathbf{G} = \begin{bmatrix} \mathbf{V}_1 \\ \mathbf{V}_2 \\ \mathbf{V}_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}_{3\times 6}$$

$\mathbf{P}_{3\times 3}$ $\mathbf{I}_{3\times 3}$

Para el bloque m=[1 1 0] , operando en modulo-2, se tiene:

$$\mathbf{mG} = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{V}_1 \\ \mathbf{V}_2 \\ \mathbf{V}_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & \underbrace{1 & 1 & 0}_{\mathbf{m}} \end{bmatrix}$$

Matriz de chequeo de paridad H

Nos permite decodificar la palabra recibida

Si G (k×n) es la matriz generadora, existe H (n-k×n) cuyas filas son ortogonales a las de G:

$$\mathbf{G} \cdot \mathbf{H}^{\mathrm{T}} = 0$$

Para un código sistemático, H debe ser:

$$\mathbf{H} = \left[\mathbf{I}_{\mathbf{n-k}} \middle| \mathbf{P}^{\mathrm{T}}\right]_{\mathbf{n-k \times n}} \qquad \mathbf{H}^{\mathrm{T}} = \begin{bmatrix} \mathbf{I}_{\mathbf{n-k}} \\ \hline \mathbf{P} \end{bmatrix}_{\mathbf{n \times n-k}}$$

Cuando se recibe una palabra código sin errores, se cumple que:

$$\mathbf{U} = \mathbf{mG} \longrightarrow \mathbf{U} \cdot \mathbf{H}^{\mathrm{T}} = \mathbf{mG} \cdot \mathbf{H}^{\mathrm{T}} = \mathbf{m}\left(\mathbf{G} \cdot \mathbf{H}^{\mathrm{T}}\right) = 0$$

Síndrome

Si $r=[r_1\ r_2\ \ldots\ r_n]$ es la palabra recibida, puede descomponerse la palabra código transmitida U más el vector de errores producidos en el canal es:

$$\mathbf{r} = \mathbf{U} + \mathbf{e}$$

Si multiplicamos por H, obtenemos el síndrome S:

$$\mathbf{S} = \mathbf{r} \cdot \mathbf{H}^T = (\mathbf{U} + \mathbf{e}) \cdot \mathbf{H}^T = \mathbf{U} \cdot \mathbf{H}^T + \mathbf{e} \cdot \mathbf{H}^T = \mathbf{e} \cdot \mathbf{H}^T$$

Por tanto:

- Si r no tiene errores → S=0
- Si r tiene errores detectables → S≠0 en alguna posición
- Si r contiene errores corregibles → S nos da información de la posición

Propiedad de los códigos de bloques lineales: existe una correspondencia biunívoca entre los patrones de error y los síndromes.

Síndrome. Ejemplo

Se envía U=[1 0 1 1 1 0] (palabra código de m=[1 1 0]),

y se recibe r=[0 0 1 1 1 0].

El primer bit se detecta erróneamente.

Calculamos el síndrome:

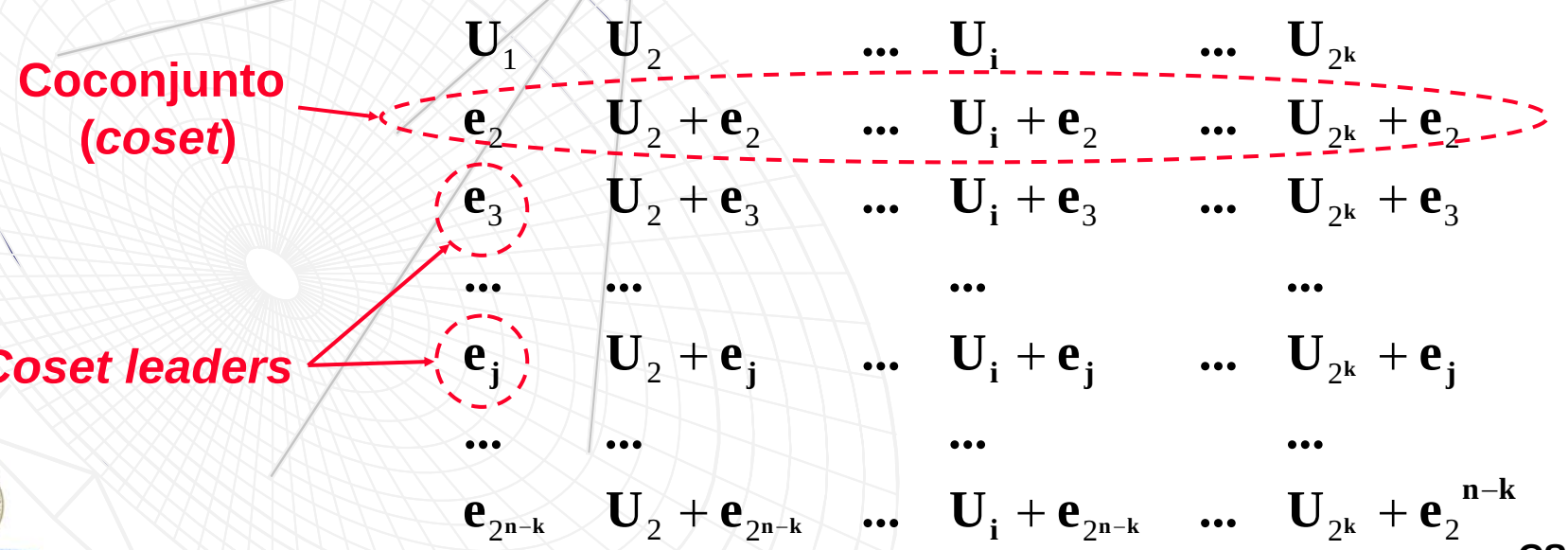
$$\mathbf{S} = \mathbf{r} \cdot \mathbf{H}^{\mathbf{T}} = \mathbf{e} \cdot \mathbf{H}^{\mathbf{T}} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

Este sistema de n-k ecuaciones lineales tiene 2^k soluciones → Hay 2^k patrones de error con el mismo síndrome

En un canal BSC, se selecciona el patrón con más ceros como error más probable

Matriz típica o estándar

- **Empezamos colocando en la primera fila los 2k vectores del código con el vector nulo U_1=(0,0...,0) como primer elemento.**
- **La primera fila está formada por todos los patrones de error corregibles**
- **De las 2^n-2^k n-tuplas restantes se toma una e_2 como vector de error y se sitúa debajo del U_1, obteniendo el resto de elementos de la fila como suma de e_2 con los vectores código de la fila primera poniendo e_2+U_i bajo el vector U_j. Para la tercera fila se coge otro que no se encuentre en las dos primeras y se le llama e_3. El resto de la fila lo forman las palabras e_3+U_i que se sitúa bajo U_j.**



Corrección de errores

Teoremas:

1) Todo código lineal de bloques (n,k) es capaz de corregir 2^{n-k} patrones de error.

2) En una fila de una matriz típica no existen dos n-tuplas iguales. Cada n-tupla aparece en una y sólo una fila.

3) Todas las 2^k n-tuplas de un coconjunto tienen el mismo síndrome. Los síndromes para diferentes coconjuntos son distintos.

4) Por lo tanto, un código bloque con distancia mínima d_{min} es capaz de detectar todos los patrones de error de d_{min} - 1 o menos errores.

5) Un código lineal (n,k) puede detectar 2^n-2^k patrones de error.

Detección y corrección de errores

1) A partir de las características del código, detectar los patrones de error corregibles.

2) Calcular la tabla de decodificación: se calculan los síndromes de todos los patrones de error anteriores.

3) Calcular el síndrome de la palabra recibida: $S=rH^T$.

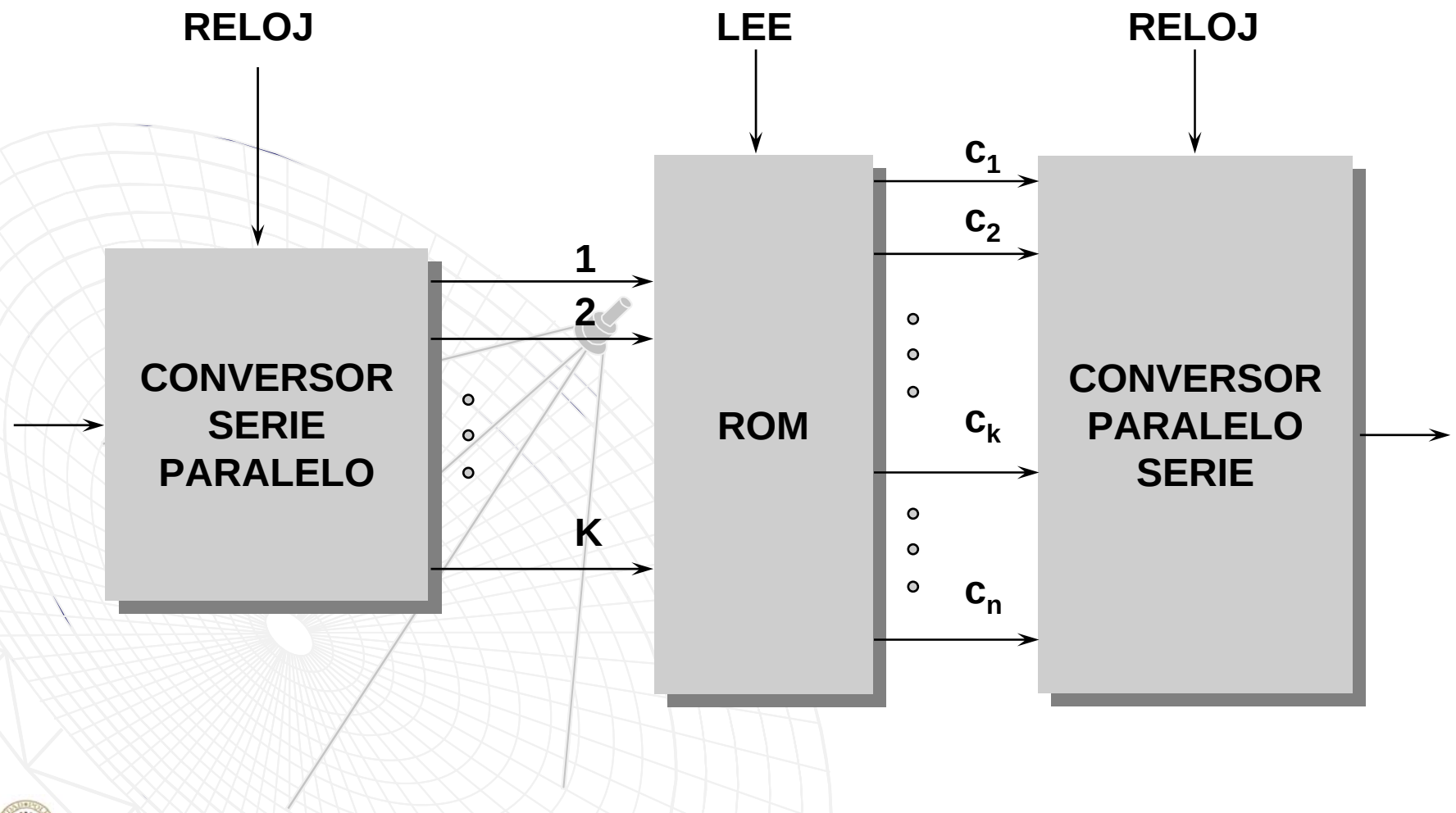
4) Localizar el coset leader (patrón de error) cuyo síndrome coincida con el calculado en 3). Éste será el patrón de errores que se ha producido en el canal e_i.

5) Decodificar: Corregir el vector recibido sumando en módulo-2 el patrón de errores encontrado: $r_{decod}=r+e_i$.

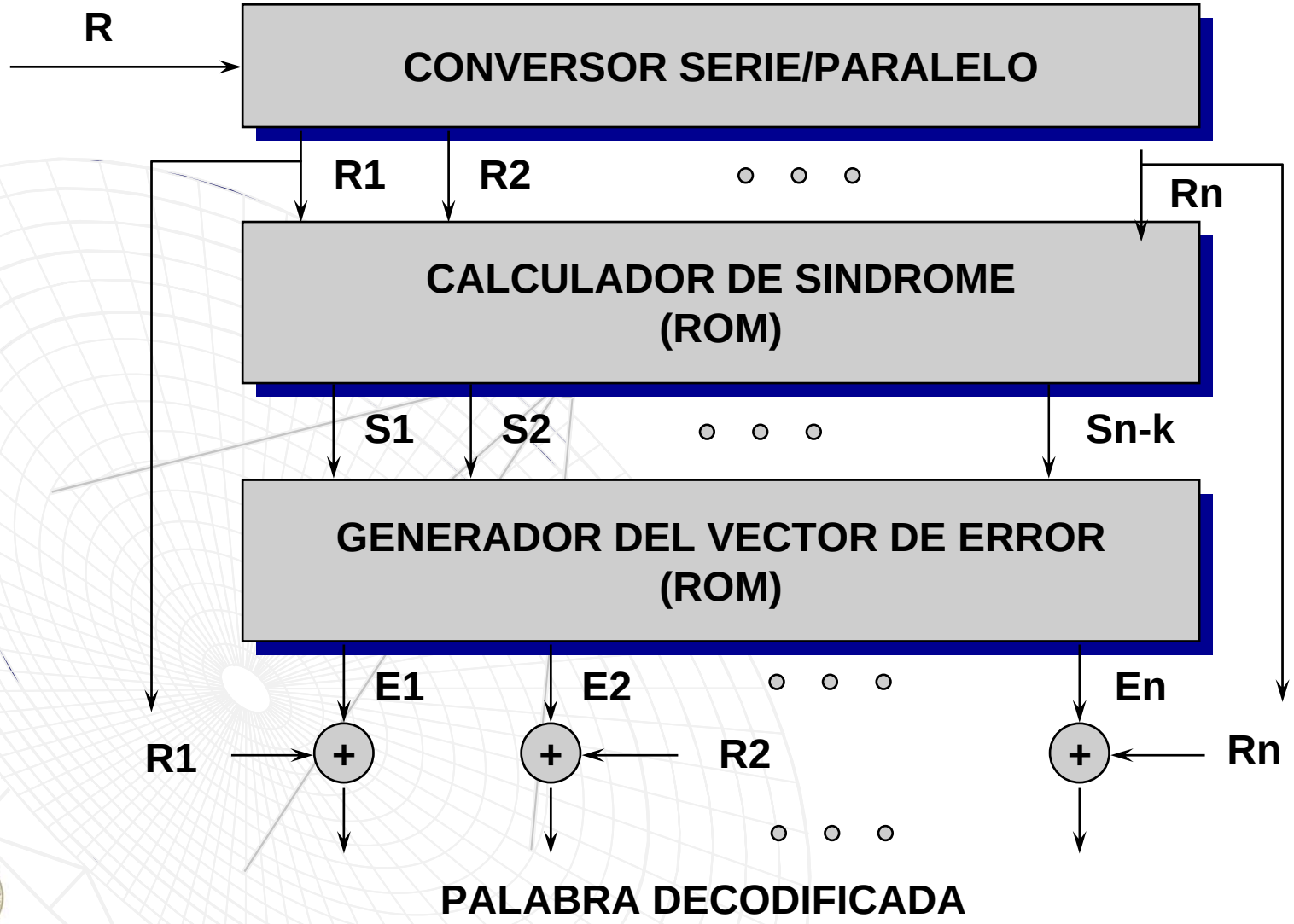
Este método, denominado decodificación de síndrome, supone un retraso en la decodificación mínimo y una probabilidad de error mínima.

No obstante, para n-k grandes, el esquema se vuelve poco práctico y puede ser inviable.

Codificador de Bloques



Decodificador de Bloques



Peso y Distancia de Hamming

Los códigos de bloques lineales forman un espacio vectorial lineal y tienen la propiedad de que dos palabras de código pueden sumarse módulo 2 y formar otra palabra de código.

Un parámetro importante de los códigos de bloques es la *distancia mínima* que determina la capacidad de corrección de errores del código.

Sea $\underline{v}$ una palabra del código de longitud n. El *peso de Hamming* (o simplemente el peso) de v, que se designa como w(v), es el número de componentes distintos de cero. Por ejemplo, el peso de $\underline{v}$ = (110101) es 4.

El conjunto de todos los pesos de Hamming de un código constituye la estructura de pesos del código.

Sean $\underline{u}$ y $\underline{v}$ dos palabras de un código (n,k). Una medida de la diferencia entre $\underline{u}$ y $\underline{v}$ es el número de elementos diferentes que tienen. Esta medida se llama *distancia de Hamming* (o simplemente distancia) entre $\underline{u}$ y $\underline{v}$ y se expresa d($\underline{u}$,$\underline{v}$).

Por ejemplo, si $\underline{v}$ = (110101) y $\underline{u}$ = (111000) entonces d($\underline{u}$,$\underline{v}$) = 3.

Claramente d(u,v) = w(u $\oplus$ v), donde u $\oplus$ v = (001101) es otra palabra del código.

Distancia Mínima (n,k,d_{min})

De la discusión anterior resulta claro que la distancia mínima d_{min} de un código es igual al peso mínimo de sus palabras de código no nulas:

$$d_{min} = w_{min} = min\, w(v), \quad v \neq 0$$

Si se usa un código lineal de bloques de distancia mínima d_{min} para corrección de errores se deseará conocer cuántos errores será capaz de corregir.

Sean v y r el código transmitido y la secuencia recibida, respectivamente. Para cualquier otra palabra de código u las distancias de Hamming entre v, r y u satisfarán la siguiente desigualdad:

$$d(v,r) + d(u,r) \geq d(v,u) \geq d_{min}$$

Supongamos que se producen t errores durante la transmisión de v. Entonces v y r diferirán en t bits y por tanto d(v,r) = t. Obtenemos así que:

$$d(u,r) \geq d_{min} - t$$

Capacidad de Corrección

En base a la regla de *decodificación de máxima verosimilitud* la secuencia recibida r será decodificada correctamente como la palabra transmitida v si la distancia de Hamming es menor que la distancia d(r,u) a cualquier otra palabra u del código.

$$d(u,r) > t$$

Por tanto, la distancia mínima d_{min} debe ser al menos 2t+1.

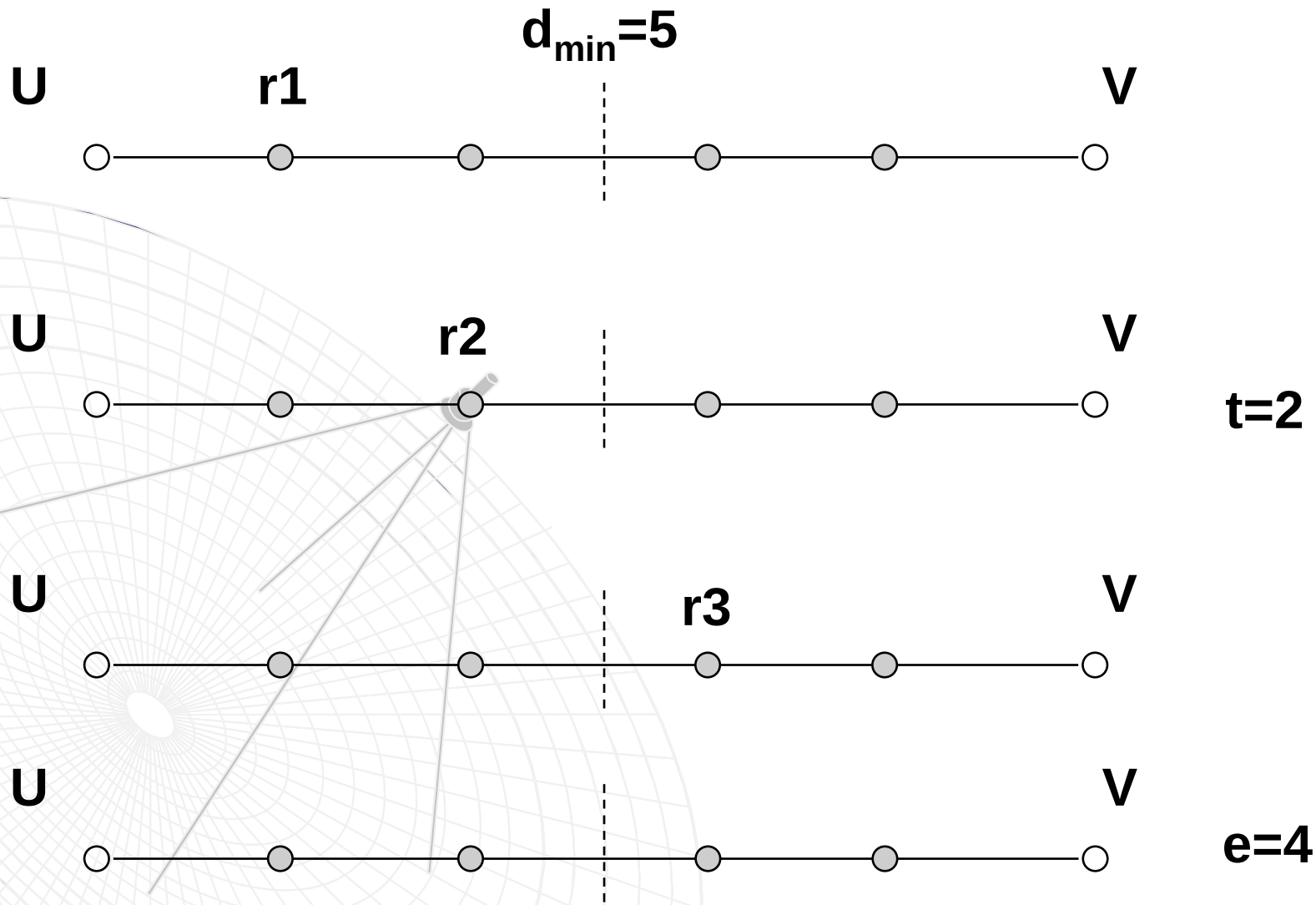
Resumiendo los anteriores resultados, un código de bloques de distancia mínima d_{min} garantiza la corrección de cualquier combinación de:

$$\mathbf{t} = \left\lfloor \frac{\mathbf{d}_{\mathbf{min}} - 1}{2} \right\rfloor$$

o menos errores, donde $\lfloor x \rfloor$ indica el mayor entero menor o igual que x.

El parámetro $t=\lfloor (d_{min}-1)/2 \rfloor$ es la *capacidad de corrección de errores* aleatorios del código.

Corrección y Detección



Número de palabras con distancia menor o igual a t

El código se dice que corrige t errores aunque, en general, es capaz de corregir también combinaciones de t+1 o incluso más errores.

De hecho, cada código de bloques lineal (n,k) es capaz de corregir 2^{n-k} combinaciones de errores, incluyendo aquellas de t o menos errores.

Los códigos que corrigen todas las combinaciones de t o menos errores y no otras se llaman *códigos perfectos*. Sin embargo, se han encontrado sólo unos pocos códigos perfectos.

Para un código (n,k), con capacidad de corregir t errores, el número de palabras con distancia menor o igual que t a una posible palabra transmitida es:

$$\sum_{i=0}^{t} \binom{n}{i}$$

Ya que hay 2^k posibles palabras de código transmitidas, el número total de palabras con distancia menor o igual que t a otras es:

$$2^k \sum_{i=0}^{t} \binom{n}{i}$$

Límites de Empaquetado

El número de secuencias de llegada no puede ser superior a 2^n. Por tanto, debe satisfacerse la desigualdad:

$$2^k \sum_{i=0}^{t} \binom{n}{i} \leq 2^n$$

o lo que es lo mismo:

$$\sum_{i=0}^{t} \binom{n}{i} \leq 2^{n-k}$$

La expresión anterior constituye el llamado *límite superior de Hamming* o límite de empaquetado esférico.

La igualdad se produce sólo cuando el código es perfecto.

Otro límite útil es el límite inferior de Varsharmov-Gilbert dado por:

$$\sum_{i=0}^{d_{min}-2} \binom{n-1}{i} > 2^{n-k}$$

Ejemplo de Empaquetado

Se desea realizar un código (63,k) con distancia mínima 5. Obtener los posibles valores de k.

Solución:

Debe cumplirse:
$$\sum_{i=0}^{t}\binom{n}{i} \le 2^{n-k} < \sum_{i=0}^{d_{min}-2}\binom{n-1}{i}$$

$$t = \left\lfloor \frac{d_{min}-1}{2} \right\rfloor = 2$$

$$\sum_{i=0}^{2}\binom{63}{i} = 2017 \quad \Rightarrow \quad \frac{\log(2017)}{\log(2)} = 10.978$$

$$\sum_{i=0}^{3}\binom{63-1}{i} = 39770 \quad \Rightarrow \quad \frac{\log(39770)}{\log(2)} = 15.28$$

Por tanto: $10.978 \le 63 - k < 15.28 \quad \Rightarrow$ **$47 \le k < 52$**

Capacidad de Detección de Errores

Sea A_j el número de palabras de código con peso j.

El conjunto $A_0, A_1, \ldots, A_n$ recibe el nombre de distribución de pesos del código.

La *probabilidad de error no detectado* en un canal BSC es:

$$P_{nd} = \sum_{i=0}^{t} A_j p^j (1-p)^{n-j}$$

donde p es la probabilidad de transición del canal.

Si la distancia mínima es d_{min} entonces los A_1 hasta A_{dmin-1} son cero.

Tasa de Error en Códigos de Bloques Lineales

Consideremos un canal binario simétrico BSC con probabilidad de transición p para el sistema sin codificación y p' con codificación, cuando se usa un código (n,k) con capacidad de corregir t errores.

Suponemos que la tasa de información R_b y la potencia de portadora C son iguales en ambos sistemas.

Sea Ec la energía por bit en el sistema codificado mientras que Eb es la energía por bit en el sistema sin codificar.

$$E_c = \frac{k}{n} E_b < E_b$$

Por tanto, la energía por bit se reduce con el uso de la codificación, con lo que aumenta la tasa de error, pero se obtiene una ganancia neta en características gracias a la capacidad de corrección de errores de la decodificación.

Tasa de Error en Códigos de Bloques Lineales

La probabilidad media de bloque erróneo sin codificación P_{wu} es:

$$P_{wu} = 1 - (1 - p)^k$$

La probabilidad media de error en una palabra de código P_w tiene como límite superior:

$$P_w \leq \sum_{i=t+1}^{n} \binom{n}{i} p'^i (1 - p')^{n-i}$$

La igualdad en la ecuación anterior se produce cuando el código es perfecto. Recuérdese que un código no perfecto puede corregir un total de 2^{n-k} combinaciones de errores incluyendo aquellas con t o menos errores.

Cuando np'<< 1 el primer término de la suma predomina sobre los demás y se puede aproximar la probabilidad de error por:

$$P_w \leq \binom{n}{t+1} p'^{t+1} (1 - p')^{n-t-1}$$

Las ecuaciones obtenidas permiten comparar las probabilidades de bloque erróneo y de palabra errónea.

Tasa de bit erróneo

Las prestaciones del sistema codificado y sin codificar pueden compararse también a nivel de probabilidad de bit erróneo. Para el sistema sin codificar la probabilidad de bit erróneo P_b es simplemente la probabilidad de transición p del canal DMC:

$$P_b = p$$

Dado que el número de bits de información de un bloque es k, la probabilidad de bit erróneo P_b estará acotada por:

- **P_w, la probabilidad de bloque erróneo, si suponemos que todos los bits de información del bloque son erróneos.**
- **P_w/k, en el caso de que sólo un bit de información es erróneo.**

En el caso de Eb/No alta, se puede suponer que las palabras recibidas erróneas son producidas por la existencia de (t+1) bits erróneos, de los cuales, en promedio, (t+1)k/n son de información. Por tanto, puede aproximarse la tasa de error en los bits de información por (límite inferior):

$$P_b' \cong \frac{t+1}{n} P_w$$

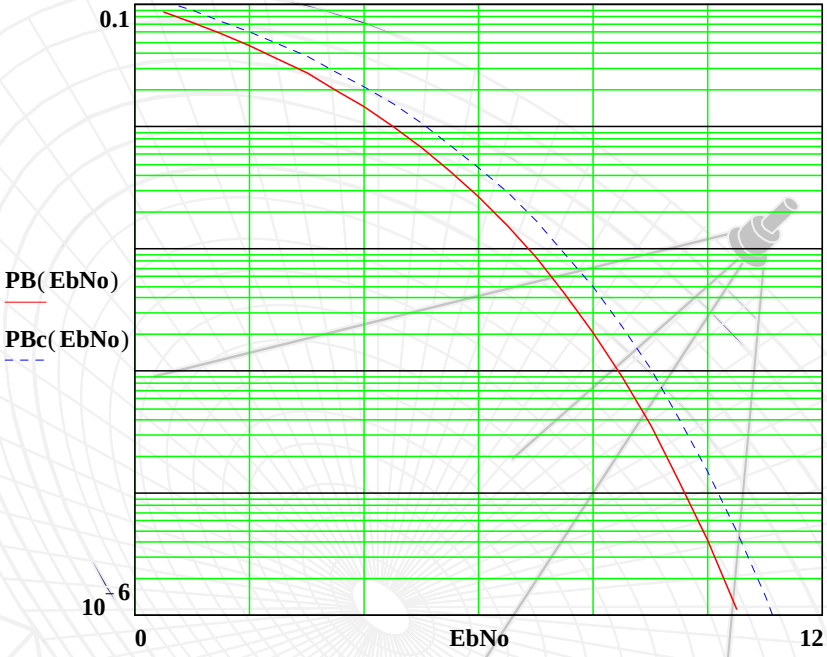
Tasa de Bit Erróneo

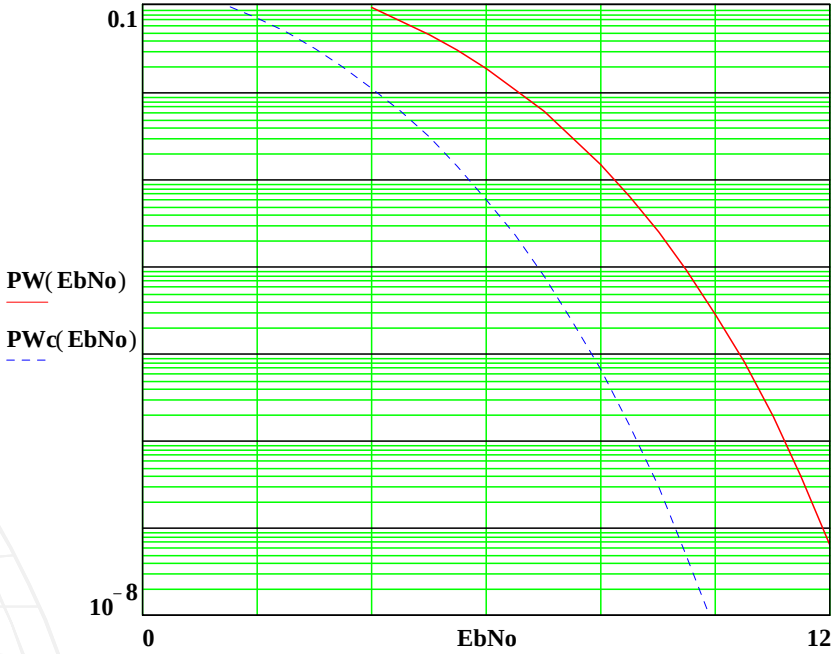
Para obtener un límite superior de la probabilidad de error en un sistema codificado P'_b suponemos que una combinación de i >t errores hace que la palabra decodificada difiera de la palabra correcta en i+t bits.

Por tanto, de los k bits de información resultarán incorrectos k(i+t)/n bits:

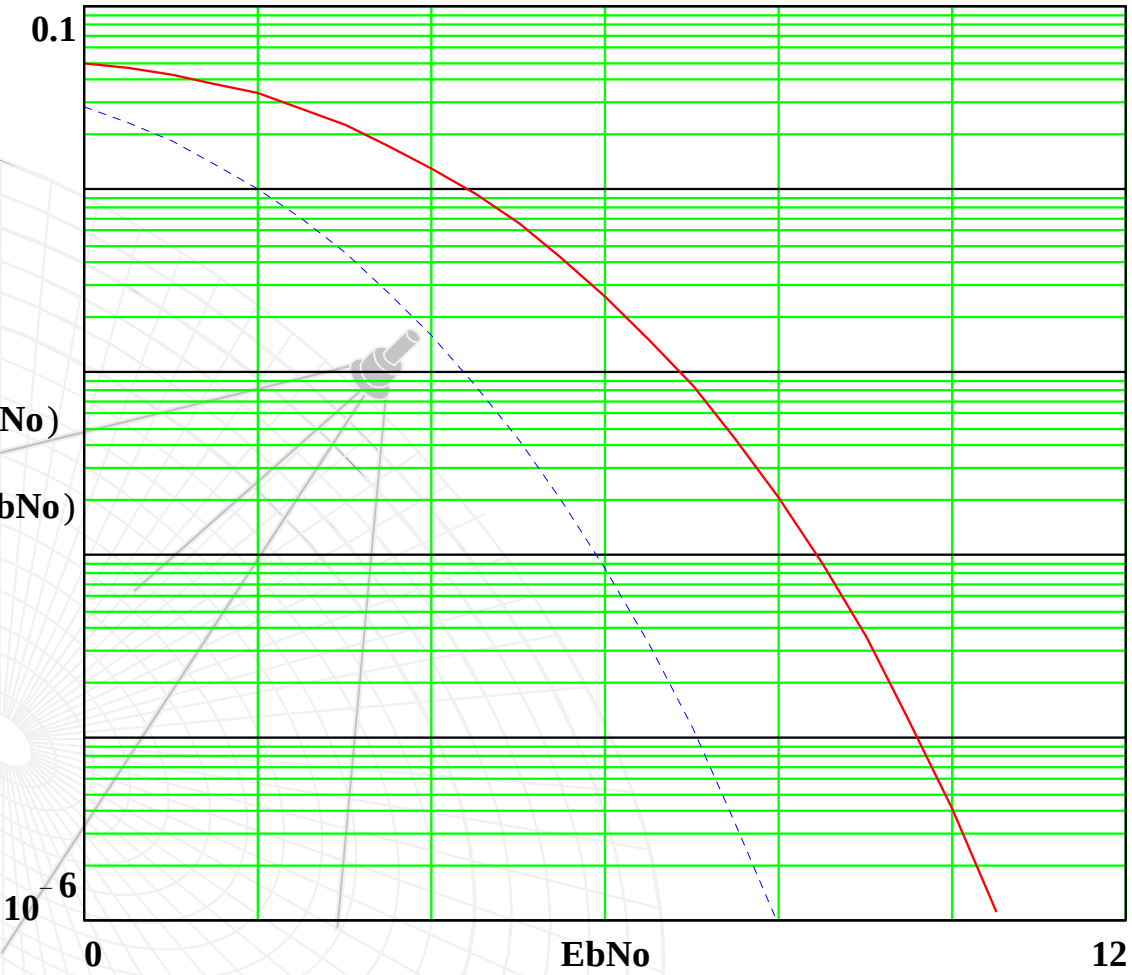
$$P'_b \leq \sum_{i=t+1}^{n} \frac{i+t}{n} \binom{n}{i} p'^{\,i} (1-p')^{n-i} \approx \frac{2t+1}{n} \binom{n}{t+1} p'^{\,t+1}$$

BER en el canal con y sin Codificación





Ganancia de Codificación



Códigos de Hamming

Su estructura es $(n,k)=(2^m-1,2^m-1-m)$,

donde m=2, 3, ... o sea (3,1), (7,4), (15,11), (31,26), ...
Su distancia mínima es $d_{min}=3$
Son capaces de corregir t=1 error
y detectar e=2 errores.

Son <u>códigos perfectos</u> y <u>sistemáticos</u>.

Veamos el código (7,4). La matriz P es:

$$P := \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

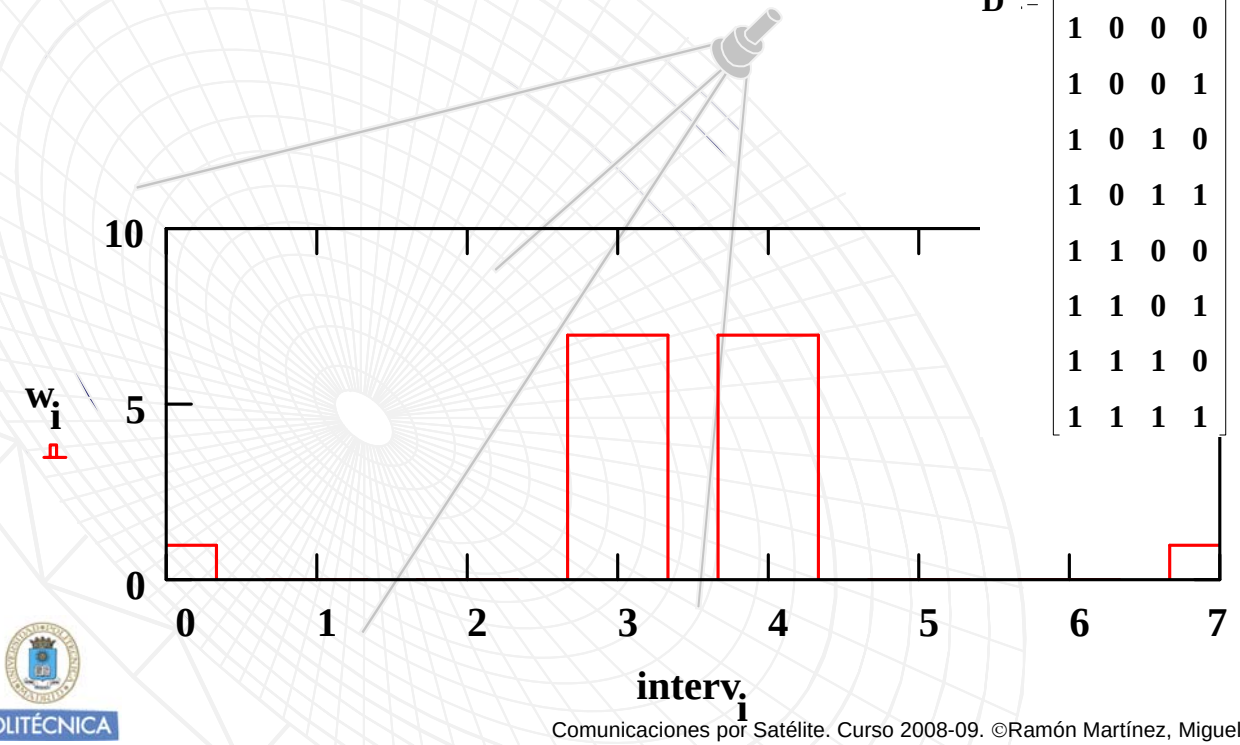
La matriz generadora de código G es:

$$G := \text{augment}(\text{identity}(4), P) \qquad G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Código de Hamming (7,4)

Las 2^4 =16 posibles secuencias de datos a codificar y las palabras codificadas son D y C. Los pesos son W.

$$D := \begin{bmatrix} 0&0&0&0 \\ 0&0&0&1 \\ 0&0&1&0 \\ 0&0&1&1 \\ 0&1&0&0 \\ 0&1&0&1 \\ 0&1&1&0 \\ 0&1&1&1 \\ 1&0&0&0 \\ 1&0&0&1 \\ 1&0&1&0 \\ 1&0&1&1 \\ 1&1&0&0 \\ 1&1&0&1 \\ 1&1&1&0 \\ 1&1&1&1 \end{bmatrix} \quad C = \begin{bmatrix} 0&0&0&0&0&0&0 \\ 0&0&0&1&0&1&1 \\ 0&0&1&0&1&0&1 \\ 0&0&1&1&1&1&0 \\ 0&1&0&0&1&1&0 \\ 0&1&0&1&1&0&1 \\ 0&1&1&0&0&1&1 \\ 0&1&1&1&0&0&0 \\ 1&0&0&0&1&1&1 \\ 1&0&0&1&1&0&0 \\ 1&0&1&0&0&1&0 \\ 1&0&1&1&0&0&1 \\ 1&1&0&0&0&0&1 \\ 1&1&0&1&0&1&0 \\ 1&1&1&0&1&0&0 \\ 1&1&1&1&1&1&1 \end{bmatrix} \quad W = \begin{bmatrix} 0 \\ 3 \\ 3 \\ 4 \\ 3 \\ 4 \\ 4 \\ 3 \\ 4 \\ 3 \\ 3 \\ 4 \\ 3 \\ 4 \\ 4 \\ 7 \end{bmatrix}$$



Código de Hamming (7,4)

La matriz de chequeo de paridad H será:

$$H := \text{augment}\left(P^T, \text{identity}(3)\right) \qquad H = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Para detectar errores calculamos el síndrome $S=RxH^T$. La posición del error está en el bit que corresponde con el número de fila de la matriz H^T que es igual al síndrome.

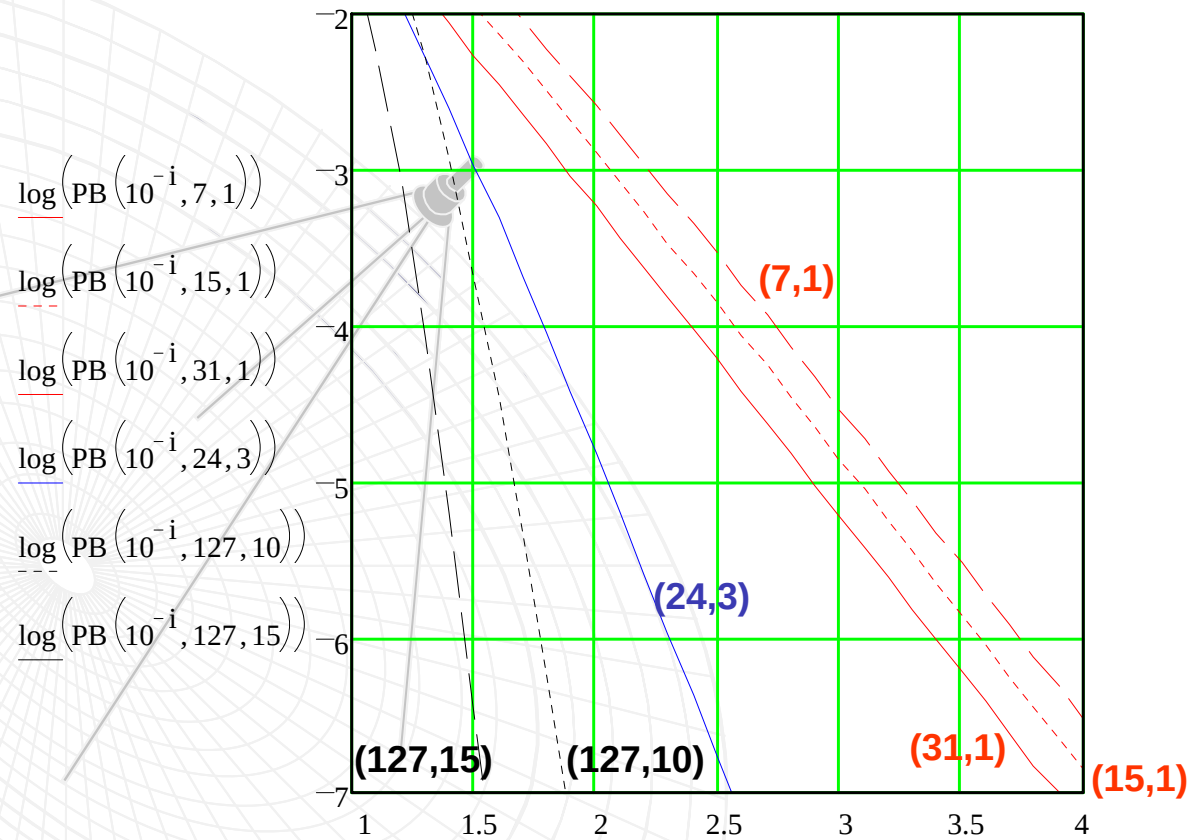
$$E = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad E \cdot H^T = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad H^T = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Los tres últimos bits son de chequeo y si el error se produce en ellos no habrá que hacer ninguna corrección. La secuencia enviada será la de los 4 primeros bits dado que el código es sistemático.

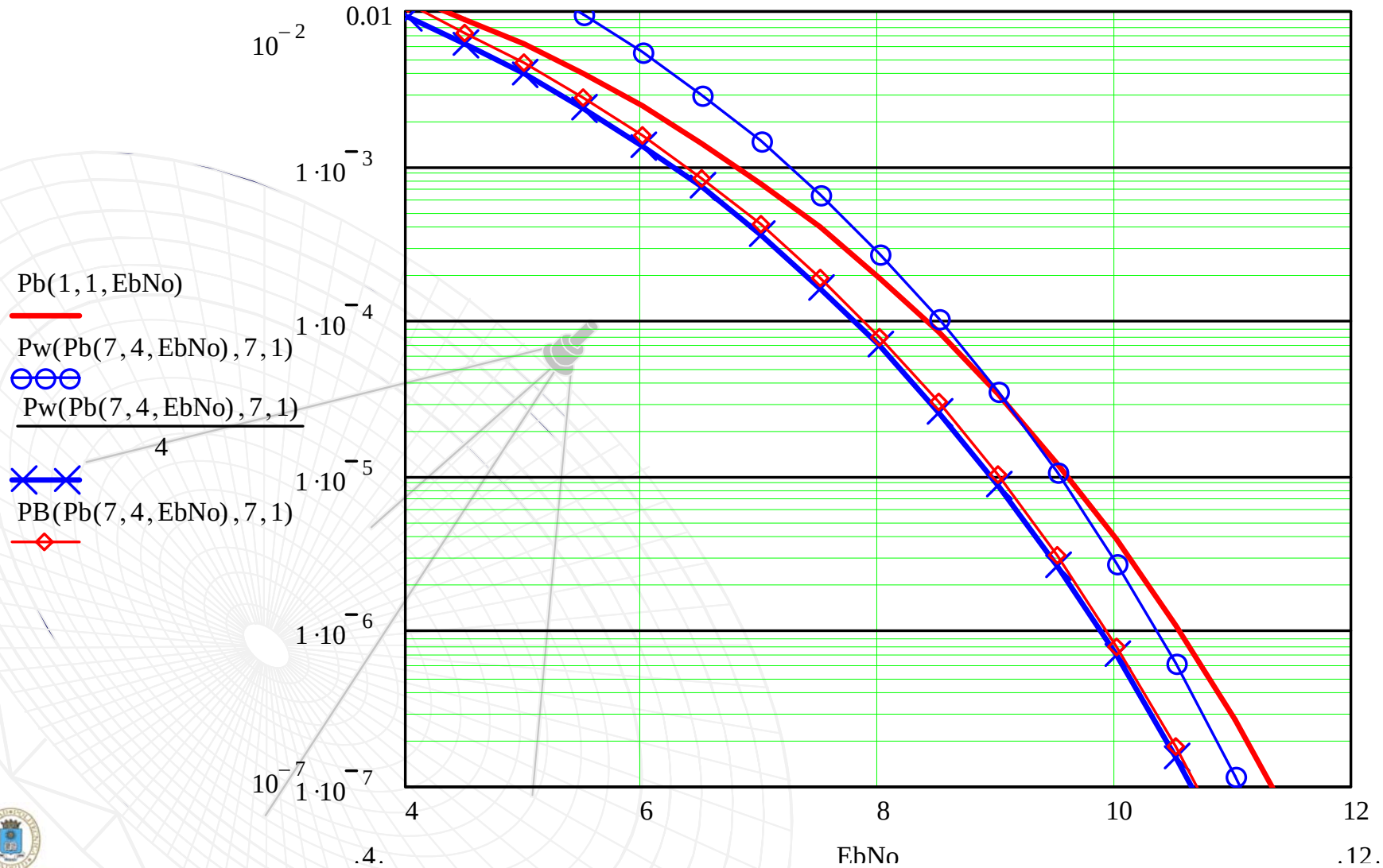
Tasa de Bit Erróneo

Para calcular la probabilidad de bit erróneo en un canal BSC (función de transferencia del código) con probabilidad de transición p aplicamos la fórmula:

$$PB(p,n,t) := \frac{1}{n} \cdot \sum_{j=t+1}^{n} j \cdot Comb(n,j) \cdot p^{j} \cdot (1-p)^{n-j}$$



Límites de la Tasa de Error para un Hamming (7,4)



Códigos Cíclicos Binarios

Son un tipo de códigos lineales de bloques cuya ventaja consiste en que su implementación puede hacerse de forma simple utilizando puertas lógicas y registros de desplazamiento.

Un *código cíclico binario* C está compuesto por palabras de código V(n,k) de la forma

$$V = (v_0, v_1, v_2, \cdots, v_{n-1})$$

tal que si la palabra de código se desplaza i-bits a la derecha la nueva palabra V^i

$$V^i = (v_{n-i}, v_{n-i+1}, \cdots, v_0, v_1, \cdots, v_{n-1-i})$$

es también una palabra del código.

Por lo tanto, en un código cíclico binario (n,k) hay n posibles palabras de código que pueden obtenerse por los n posibles desplazamientos.

La palabra de código V puede representarse por un polinomio de orden n-1 en el que los coeficientes v_0, v_1, etc. pueden valer 0 o 1:

$$V(x) = v_0 + v_1 x + \cdots v_{n-1} x^{n-1}$$

Códigos Cíclicos Binarios

El vector V^i desplazado i bits será:

$$V^i(x) = v_{n-i} + v_{n-i+1}x + \cdots + v_{n-i-1}x^{n-1}$$

y puede obtenerse como el resto de la división de $x^i V(x)$ por x^n+1, o sea:

$$x^i V(x) = q(x)(x^n + 1) + V^i(x)$$

donde q(x) es el polinomio cociente.

Las palabras de código pueden generarse de <u>dos formas</u>:

1) Formamos palabras de <u>código no sistemático</u> usando un polinomio generador g(x) tal que, a partir de un polinomio de datos D(x), formamos: V(x) = D(x) g(x) , donde:

$$D(x) = d_0 + d_1 x + \cdots + d_{k-1}x^{k-1} \qquad g(x) = g_0 + g_1 x + \cdots + g_{n-k}x^{n-k}$$

Códigos Cíclicos Binarios

2) Un procedimiento alternativo para formar un <u>código sistemático</u> es usar g(x) para generar un código de la forma: $V(x) = r(x) + x^{n-k} D(x)$ donde r(x) es el <u>polinomio de control de paridad</u> que se obtiene:

a) multiplicando D(x) por x^{n-k},

b) dividiendo x^{n-k} D(x) por g(x),

c) el resto es r(x):

$$\frac{x^{n-k} D(x)}{g(x)} \rightarrow q(x); r(x)$$

Estas operaciones se implementan fácilmente mediante circuitos lógicos.

Ejemplo de Código (7,4)

Se quiere generar un código cíclico (7,4) con un polinomio generador g(x)=1+x+ +0x²+x³. Determinar la palabra de código correspondiente al vector de datos D=(1010) cuyo polinomio correspondiente es D(x)=1+0x+x²+0x³.

1) Código <u>no sistemático</u>:

$$V(x) = D(x)g(x) = (1+x^2)(1+x+x^3) = 1+x+x^3+x^2+x^3+x^5$$

$$= 1+x+x^2+x^5 \Rightarrow \text{V=(1110010)}$$

2) Código <u>sistemático</u>: $x^{n-k}D(x) = x^3(1+x^2) \Rightarrow \dfrac{x^{n-k}D(x)}{g(x)} = \dfrac{x^3+x^5}{1+x+x^3}$

$0 + 0x + 0x^2 + x^3 + 0x^4 + x^5$ | $1 + x + x^3$

$x^2 + x^3 \quad + x^5$ | x^2

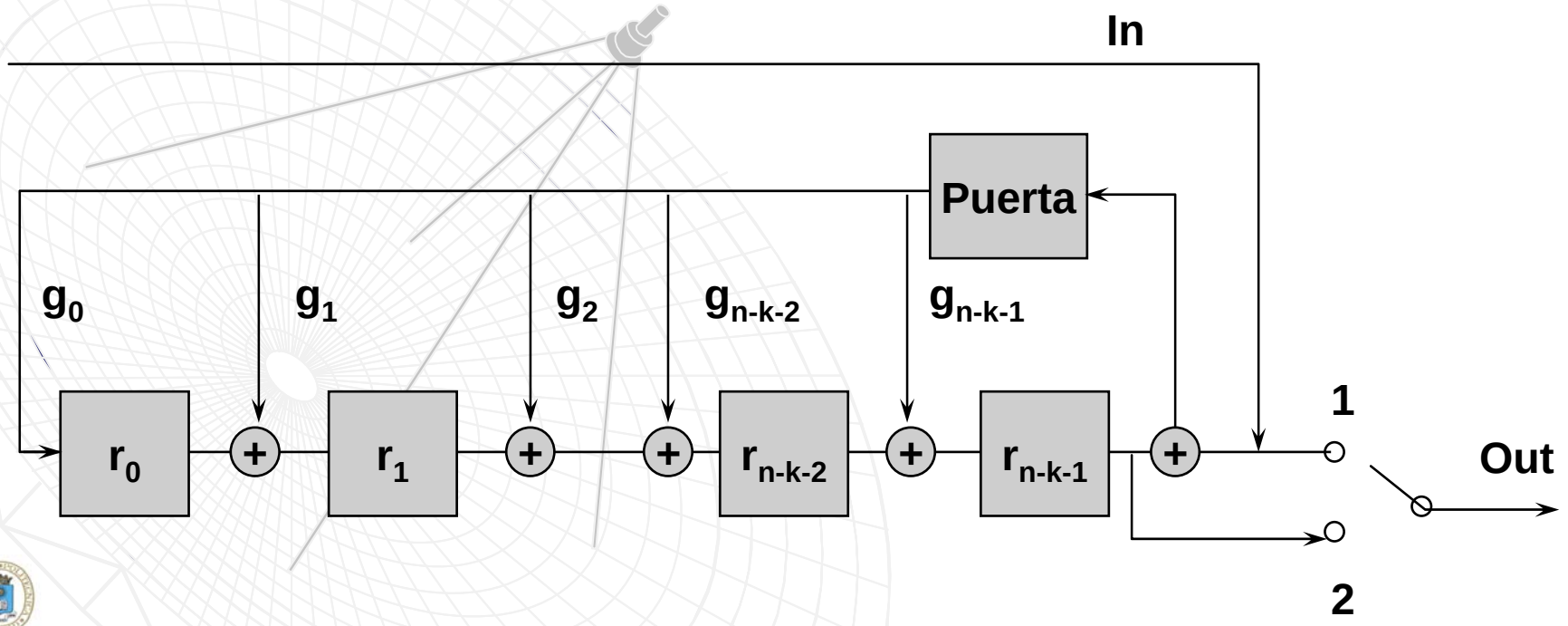
$x^2 + 0 \quad + 0$ | $q(x)=x^2$, $r(x)=x^2$

$$V(x) = x^3(1+x^2) + x^2 = x^2 + x^3 + x^5$$

V=(0011010)

Implementación de un Código Cíclico Sistemático

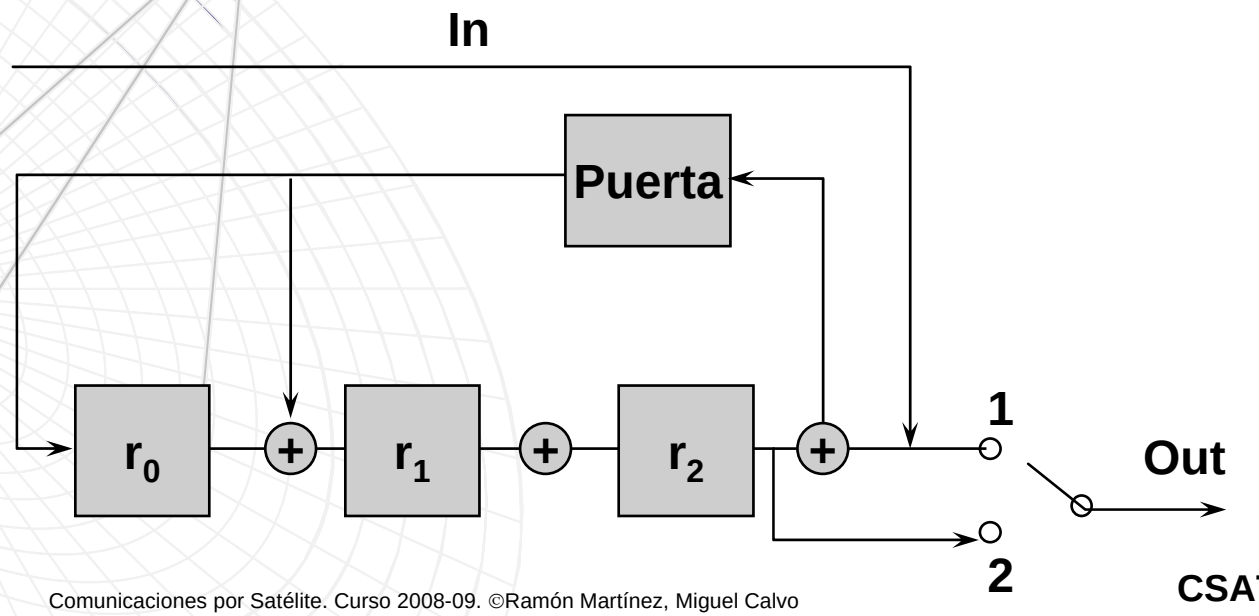
- **El vector de datos es transmitido primero con el conmutador en 1. La puerta permanece abierta durante este tiempo y los datos se van introduciendo en los registros de desplazamiento.**
- **Las conexiones generan g(x) y se realiza la operación de división.**
- **Se cierra a continuación la puerta, el conmutador se coloca en 2 y se transmite entonces el resto de la división, que es el contenido de los registros.**



Ejemplo: Hamming (7,4)

D	r_0	r_1	r_2
	$(D+ r_2)$	$(D+ r_2)+ r_0$	r_1
	0	0	0
0	0	0	0
1	1	1	0
0	0	1	1
1	0	0	1

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



Detección y Corrección de Errores

La detección de errores se realiza calculando el síndrome del vector recibido R(x).

El síndrome se calcula dividiendo el vector recibido por el polinomio generador y quedándonos con el resto S(x):

$$R(x) = q(x)g(x) + S(x)$$

Si el error es E(x) será : R(x) = V(x) + E(x) = D(x) g(x) + E(x). Por lo tanto:

$$\frac{R(x)}{g(x)} = q(x) + \frac{S(x)}{g(x)} = D(x) + \frac{E(x)}{g(x)} \Rightarrow$$

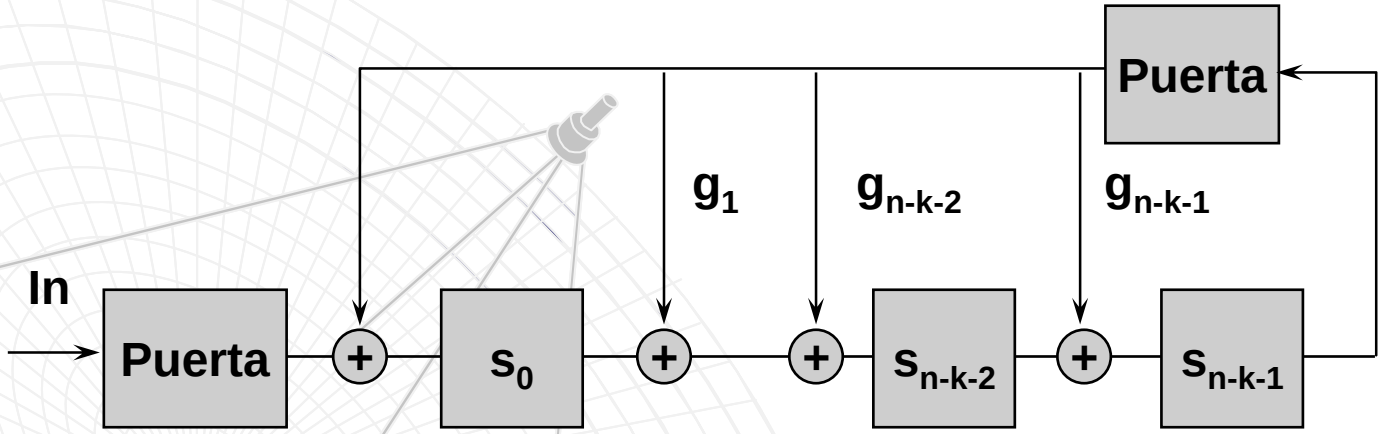
$$\Rightarrow E(x) = S(x) + \left[q(x) + D(x)\right]g(x)$$

Si el síndrome es cero, no hay error.

El síndrome proporciona además la posición del error y permite corregirlo.

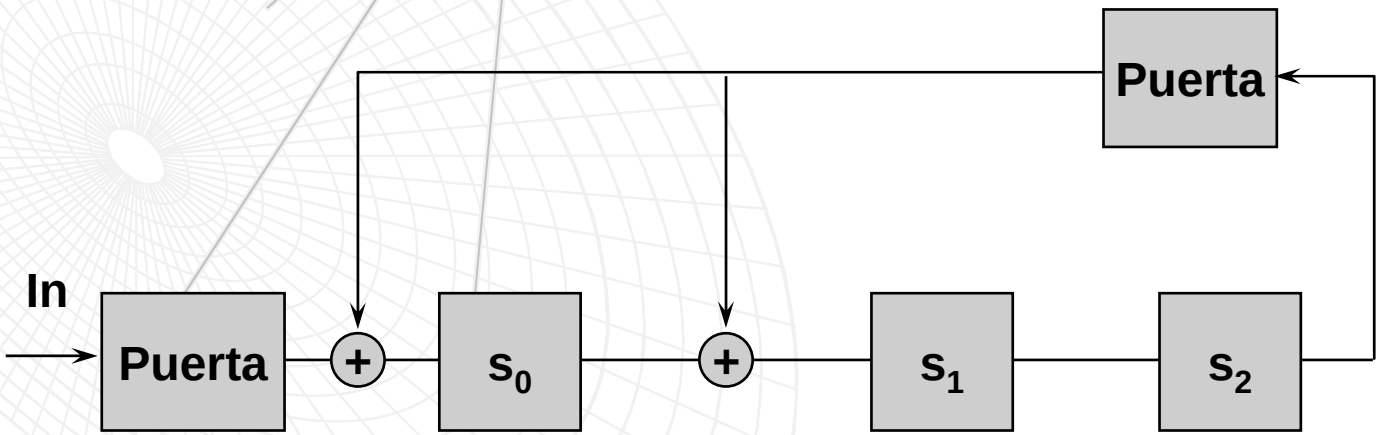
Implementación del Cálculo del Síndrome

El cálculo del síndrome se realiza con el circuito de la figura.



Ejemplo

M	s0	s1	s2
	(s2+M)	(s0+s2)	(s1)
	0	0	0
0	0	0	0
1	1	0	0
0	0	1	0
1	1	0	1
1	0	0	0
0	0	0	0
0	0	0	0



Código de Golay

Es un código (23,12) con una distancia mínima $d_{min}=7$ capaz de corregir 3 o menos errores.

Es un código perfecto.

Como $1 + x^{23} = (1 + x)g_1(x)g_2(x)$ siendo:

$$g_1(x) = 1 + x^2 + x^4 + x^5 + x^6 + x^{10} + x^{11}$$

$$g_2(x) = 1 + x + x^5 + x^6 + x^7 + x^9 + x^{11}$$

puede utilizarse bien g_1 o bien g_2 como polinomio generador.

En realidad, el código de Golay más utilizado es el extendido (24,12) que se obtiene añadiendo un bit redundante extra para así tener una tasa de codificación de exactamente ½, y aumentar la distancia mínima de 7 a 8.

Códigos Cíclicos de Chequeo de Redundancia CRC

Bajo ciertas condiciones, tales como interferencia transitoria o interferencia cocanal producida por multitrayecto, los errores tienden a presentarse en ráfagas en lugar de en forma aleatoria aislada. Hay códigos especiales para corregir estos errores, que se presentan en bits consecutivos, llamados códigos CRC.

El número de bits de chequeo de paridad que se necesitan para corregir ráfagas de q errores consecutivos es: $n-k \geq q$.

Códigos	$g(x)=(1+x)\cdot g'(x)$
CRC-12	$1 + x + x^2 + x^3 + x^{11} + x^{12}$
CRC-16	$1 + x^2 + x^{15} + x^{16}$
CRC-CCITT	$1+ x^5 + x^{12} + x^{16}$

Códigos BCH

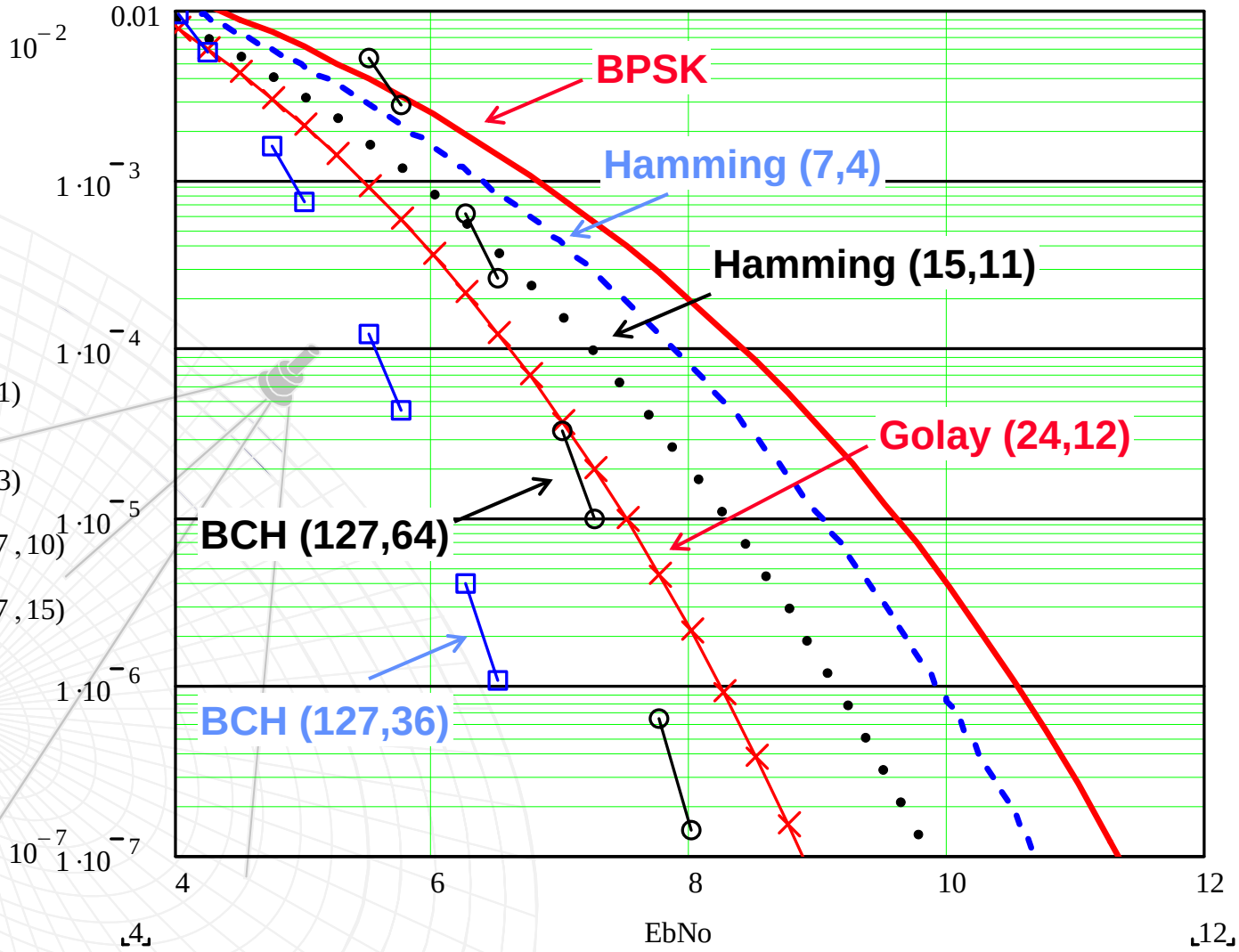
Son una generalización de los códigos de Hamming que permiten una corrección de errores múltiples. Para cualquier entero positivo m>3 y $t<2^m-1$ hay un código BCH con los siguientes parámetros:

Longitud del código: $n=2^m-1$
Longitud del bloque de información: k>n-mt
Distancia mínima: $d_{min} \geq t+1$
Capacidad de corrección: t bits

Algunos de estos códigos se recogen en la tabla adjunta.

n	k	t	m	Pol. generador
7	4	1	3	1011
15	11	1	4	10011
15	7	2	4	111010001
15	5	3	4	10100110111
31	26	1	5	100101
....				
31	6	7	5	1.1001011011110101000100111

Tasa de Bit Erróneo para códigos de Hamming, Golay y BCH



Códigos Reed-Solomon

Son un tipo importante de códigos BCH no binarios con los siguientes parámetros:

Longitud de símbolo: m bits por símbolo
Longitud de código: $n=2^m-1$ símbolos
Longitud del bloque de información: $k=n-2t$ símbolos
Distancia mínima: $d_{min}=2t+1$ símbolos
Capacidad de corrección de errores: t símbolos

Dada su capacidad de corrección de símbolos son especialmente adecuados para la corrección de errores en ráfaga. Se decodifican mediante el algoritmo de Berlekamp-Massey.

Suelen ir concatenados con un codificador convolucional (satélite).

Funcionan bien con grandes bloques de información: los símbolos de entrada son elementos de un campo de Galois $GF(q) \equiv GF(2^m)$.

Códigos Reed-Solomon

Los pasos para la construcción de un código RS son:

1) Definición y construcción de GF(2^m). Elementos y operaciones
- **Polinomio generador**
- **Generación del campo a partir de la primitiva del campo**

2) Codificación:
- **Circuito lógico típico de un código cíclico (sistemáticos)**
- **Operaciones en GF(2^m)**

3) Decodificación. Algoritmo iterativo de Berlekamp-Massey:
- **Obtener los síndromes de la palabra recibida**
- **Calcular el polinomio de localización de errores**
- **Encontrar sus raíces (posición de los errores)**
- **Corregir los errores**
- **Si hay más de t errores, eliminar los símbolos de paridad**

Códigos Reed-Solomon para satélite

En TV digital por satélite (EN 300 421) se utiliza el código Reed-Solomon:

RS (204,188, T = 8)

versión acortada del original:

RS(255,239, T = 8).

Se aplica sobre cada paquete de transporte de 188 bytes.

• Polinomio generador de código: $g(x) = (x+\lambda^0)(x+\lambda^1)(x+\lambda^2)\ldots(x+\lambda^{15})$, siendo λ = 02HEX (*primitiva* de GF(2^8)).

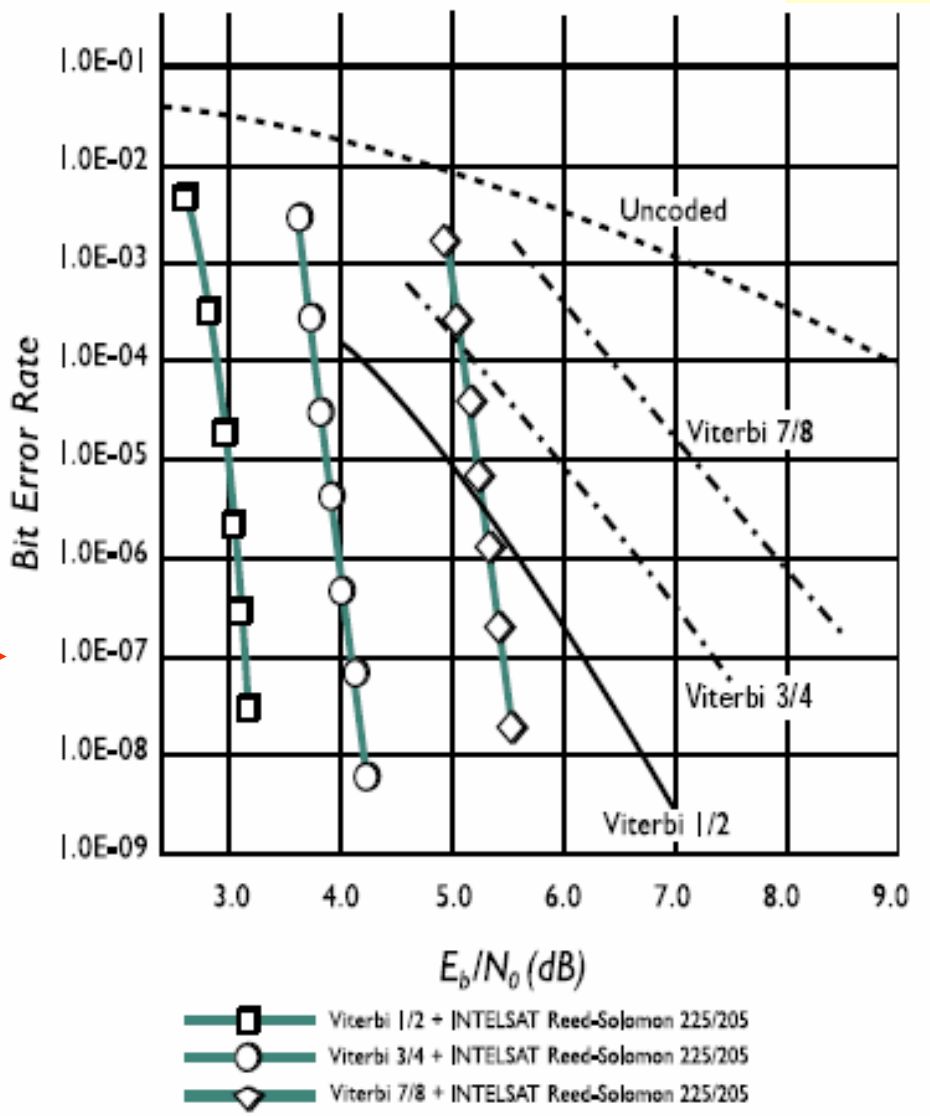
• Polinomio generador de campo GF(2^8): $p(x) = x^8 + x^4 + x^3 + x^2 + 1$ (*polinomio irreducible*).

El código Reed-Solomon acortado puede implementarse añadiendo 51 bytes a cero a los 188 bytes de información. Se introduce el bloque de 239 bytes en el codificador original RS (255,239) y se descartan posteriormente de la palabra codificada.

Reed Solomon + Convolucional

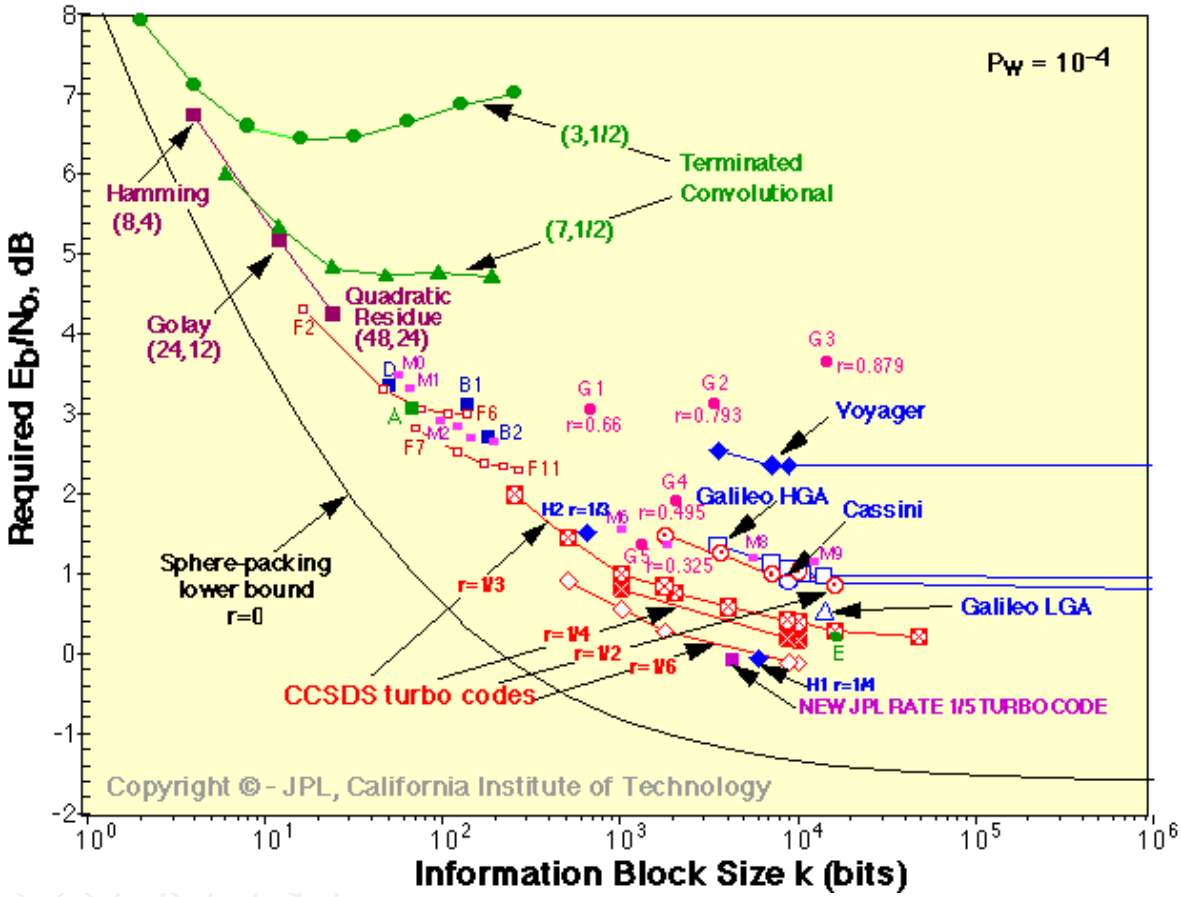
Fuente: Radyne

Ejemplo:
BER=10^{-7}
EbNo gain ~ 3 dB



RS-Misiones de espacio profundo

Voyager (Venus)
Mars Pathfinder (Marte)
Galileo (Jupiter)
Mars Exploration Rover (Marte)
Cassini (Saturno)

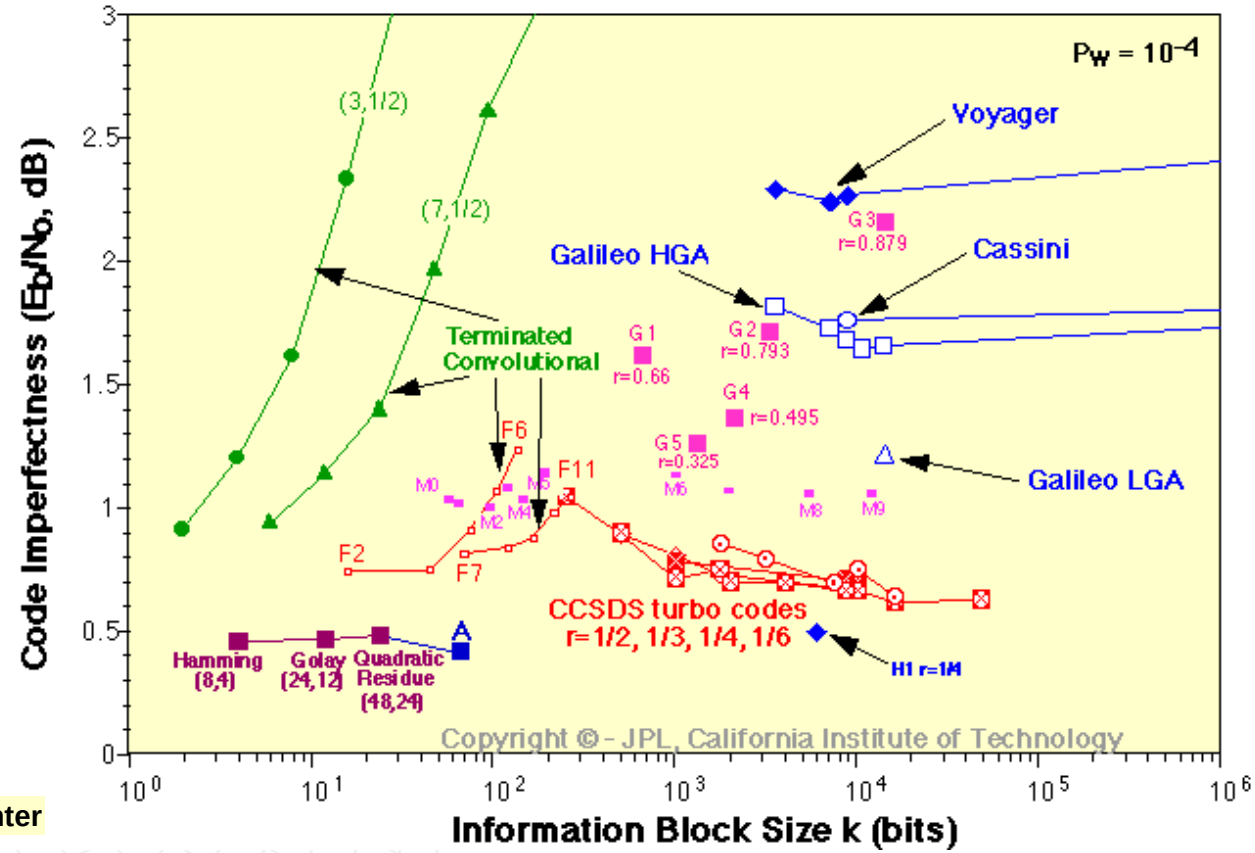


Fuente: NASA Deep Space Center

RS-Misiones de espacio profundo

Voyager (Venus)
Mars Pathfinder (Marte)
Galileo (Jupiter)
Mars Exploration Rover (Marte)
Cassini (Saturno)

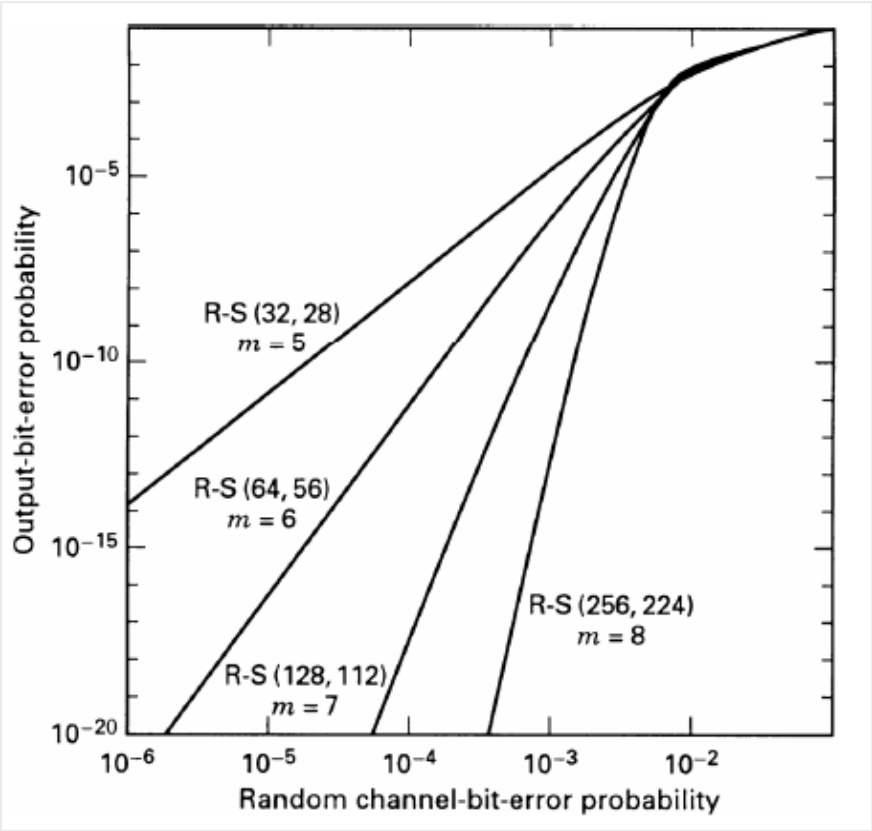
Code imperfectness of a given code is defined **as the difference between the code's required E**b/No to attain a given word error probability (Pw), and the minimum possible Eb/No required to attain the same Pw, as implied by the sphere-packing bound for codes with the same block size *k* and code rate *r*.



Fuente: NASA Deep Space Center

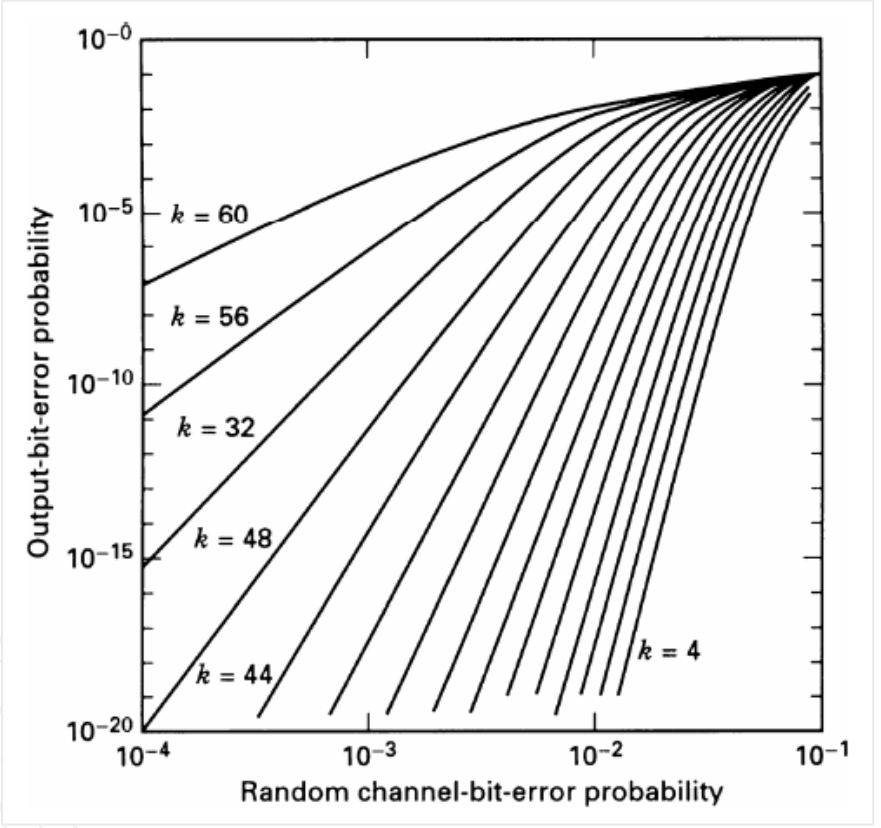
Prestaciones de códigos Reed-Solomon

RS de tasa 7/8



Prestaciones en función del alfabeto (*m*)

RS (64,k)



“Función de transferencia”

Fuente: SKLAR

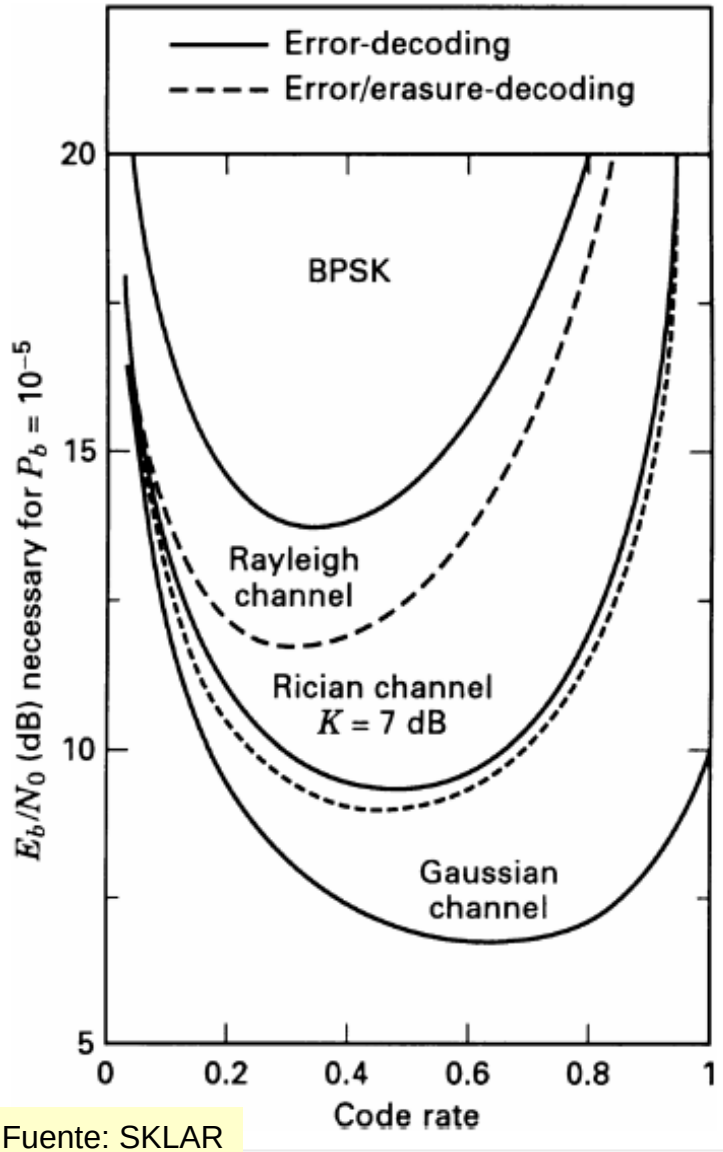
Prestaciones de códigos Reed-Solomon

Prestaciones de RS (31,k) en canales reales con BPSK

- Aparece una tasa de codificación óptima
  - 0.3 para canal Rayleigh
  - 0.5 para canal Rice (K=7 dB)
  - 0.6-0.7 para canal AWGN

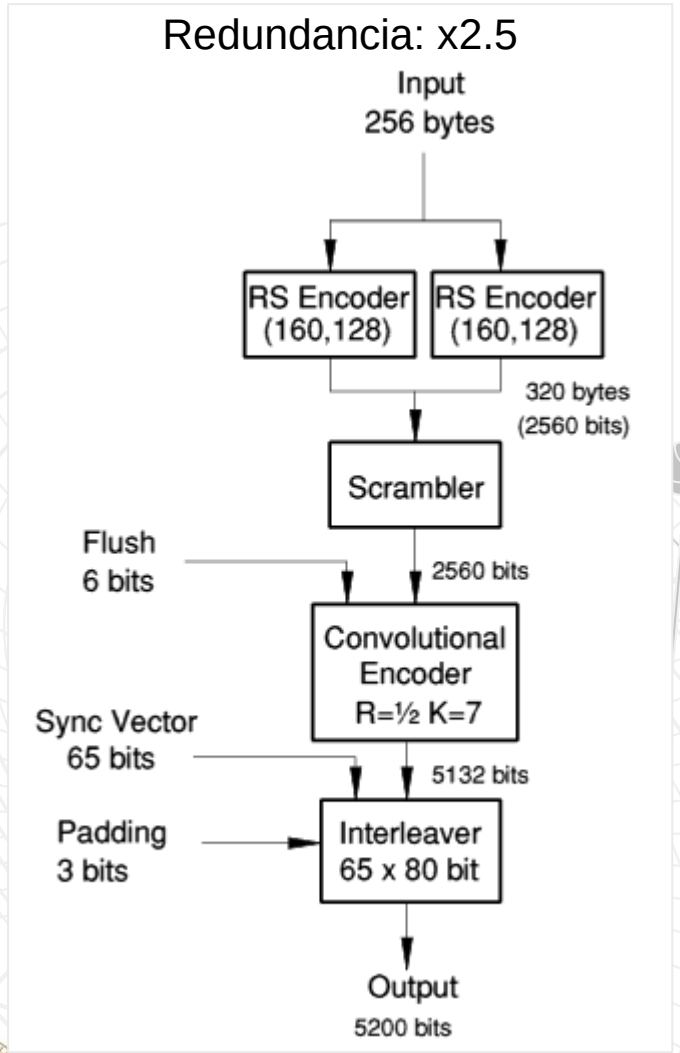
- Límite inferior: menor energía por símbolo por incluir una gran redundancia

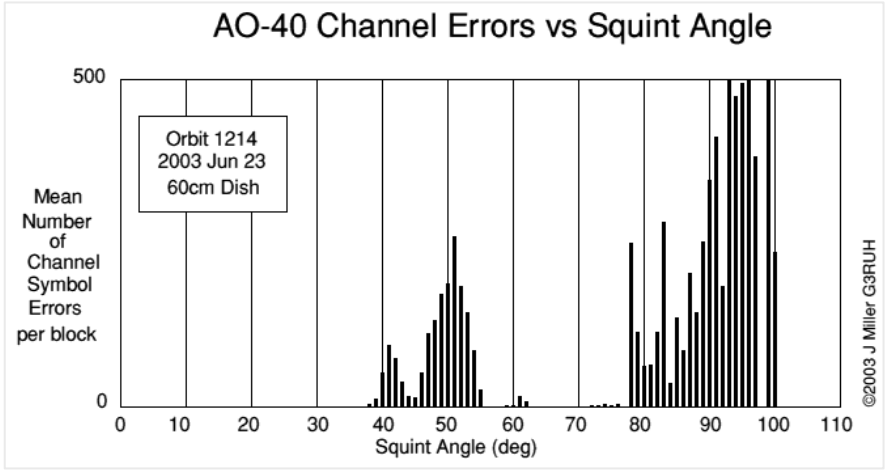
- Límite superior: menor redundancia, luego menor protección

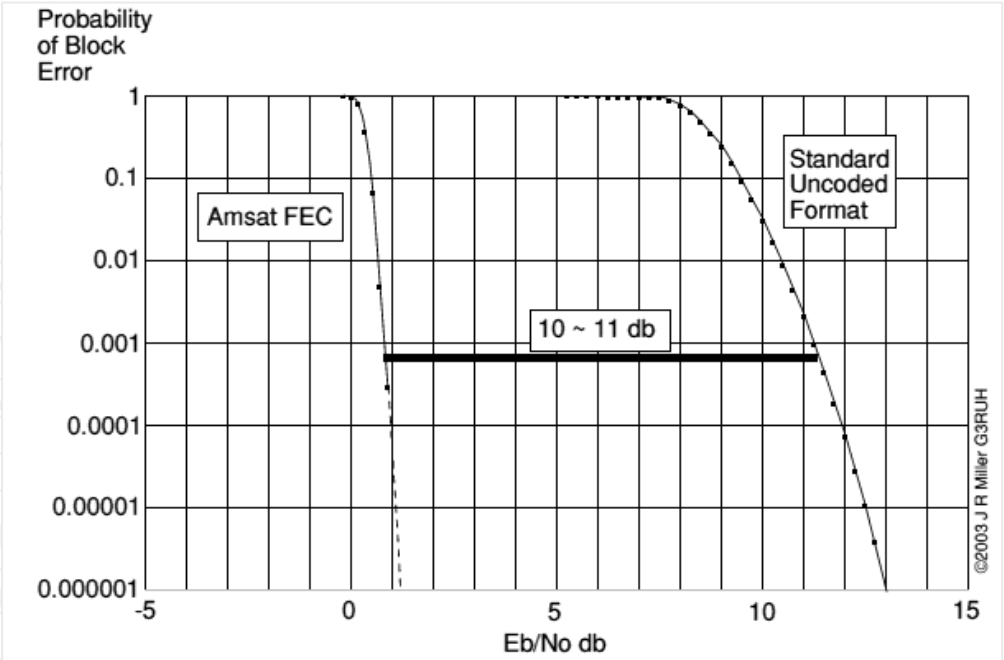


Fuente: SKLAR

OSCAR-40: FEC para telemetría







Fuente: www.amsat.org/amsat/articles/g3ruh/125.html

OSCAR-40: FEC para telemetría

Small antenna
RHCP patch
G= 8.5 dBi, θ_{-3dB}=90º

Ultra small antenna
Monopolo sobre plano de masa

Fuente: www.amsat.org/amsat/articles/g3ruh/125.html

Códigos acortados

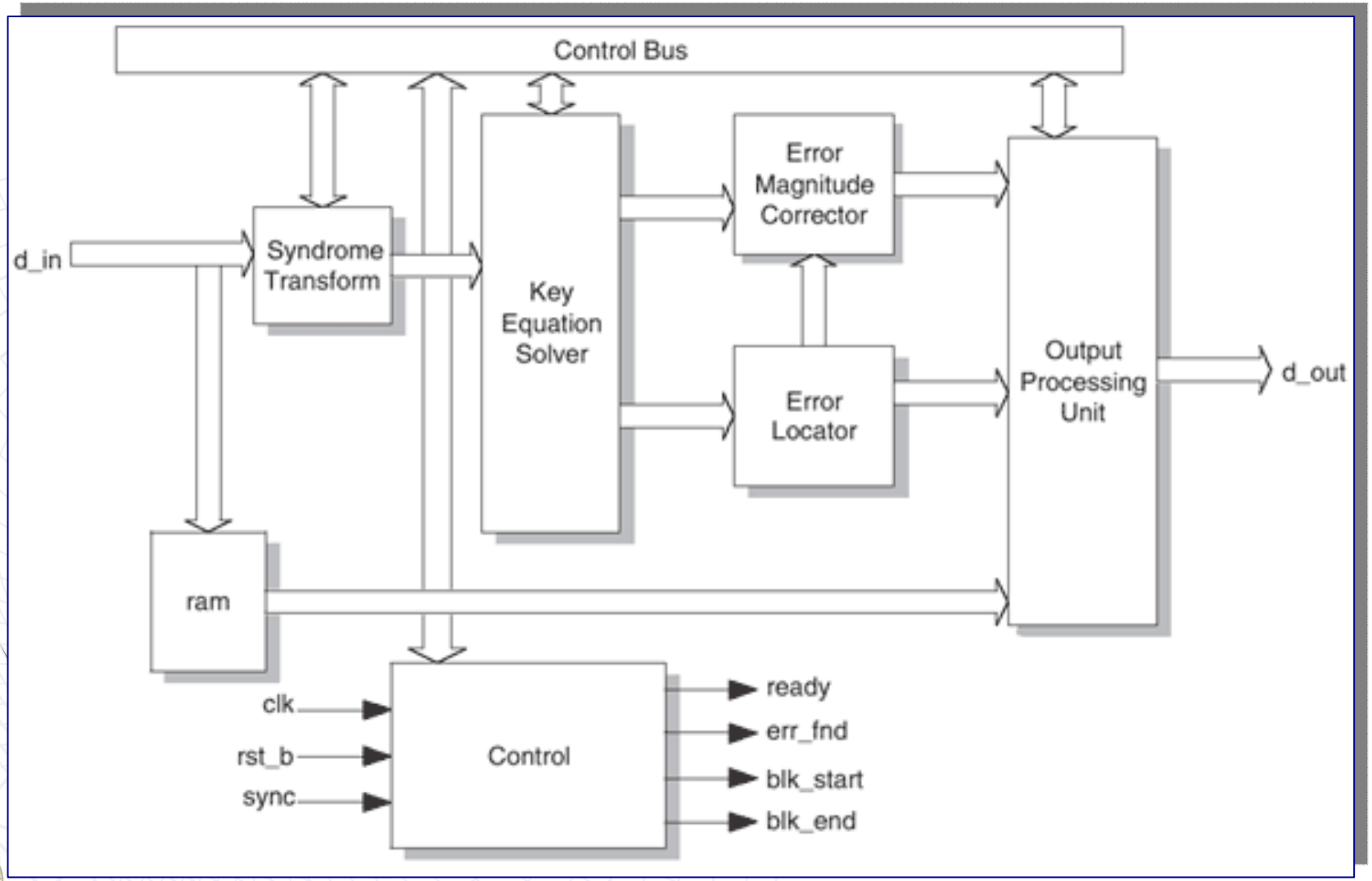
- **Se usan cuando la longitud de un código no puede utilizarse (longitud de paquetes, tasa binaria, etc.).**

- **<u>Ejemplo</u>: se desea diseñar un código de detección de errores sobre paquetes de 64 bits. La máxima redundancia permitida es de 72 bits.**
- **<u>Solución</u>: 64+8=72 no es de la forma 2^m-1, así que podríamos usar ninguno de los códigos cíclicos estudiados.**
- **Se podría usar un código Hamming (127,120,3) y acortarlo hasta llegar a k=64 $\rightarrow$ *shortened* Hamming (71,64,3)**
- **Se construyen dejando a cero s bits de la palabra código, obteniendo un código (n-s, d-s, d_s).**

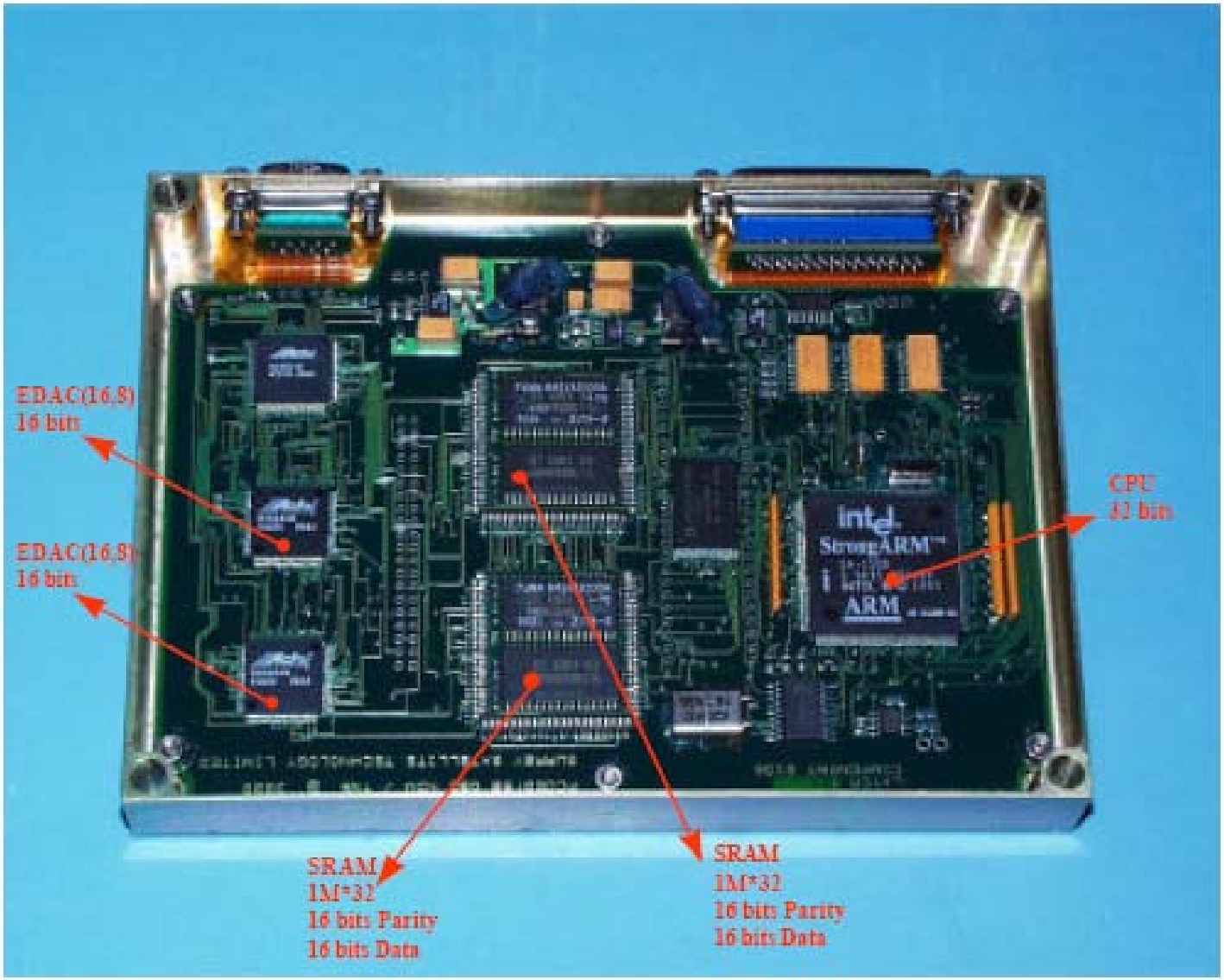
Implementación

- **Plataformas basadas en FPGAs y ASICs**
  - Operaciones lineales (sumas y productos)
  - Programación en VHDL
  - Existen IP cores
- **Software en C (alto nivel)**
- **Retardo en la codificación vs retardo de acceso a memoria**
- **Memoria requerida para las tablas de consulta (LUTs)**

Reed Solomon Decoder (Lattice)



Ejemplo de implementación (Univ. Surrey)



Plataforma para RS (Xilinx)

XILINX® LogiCORE

Reed-Solomon Decoder v5.1

DS252 April 28, 2005 — **Product Specification**

Features

- High-speed, compact Reed-Solomon Decoder
- Available for all Virtex™, Virtex-E, Virtex-II, Virtex-II Pro, Virtex-4, Spartan™-II, Spartan-IIE, Spartan-3, and Spartan-3E FPGA devices
- Implements many different Reed-Solomon coding standards
- Fully synchronous design using a single clock
- Supports continuous input data with no gap between code blocks
- Symbol size from 3 to 12 bits
- Code block length variable up to 4095 symbols
- Code block length can be dynamically varied on a block by block basis
- Supports shortened codes
- Supports error and erasure decoding
- Supports puncturing (as in IEEE802.16d)
- Supports multiple channels
- Parameterizable number of errors corrected
- Supports any primitive field polynomial for a given symbol size

software will remove the logic driving them, ensuring that FPGA resources are not wasted.

A representative symbol, with the signal names, is shown in Figure 1 and described in Table 1.

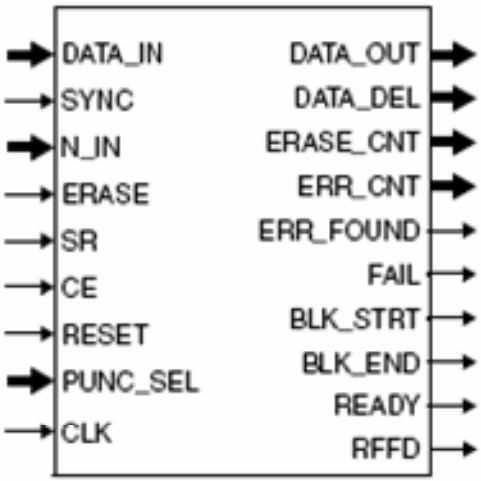
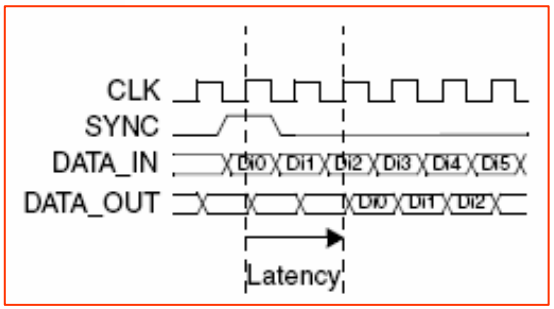
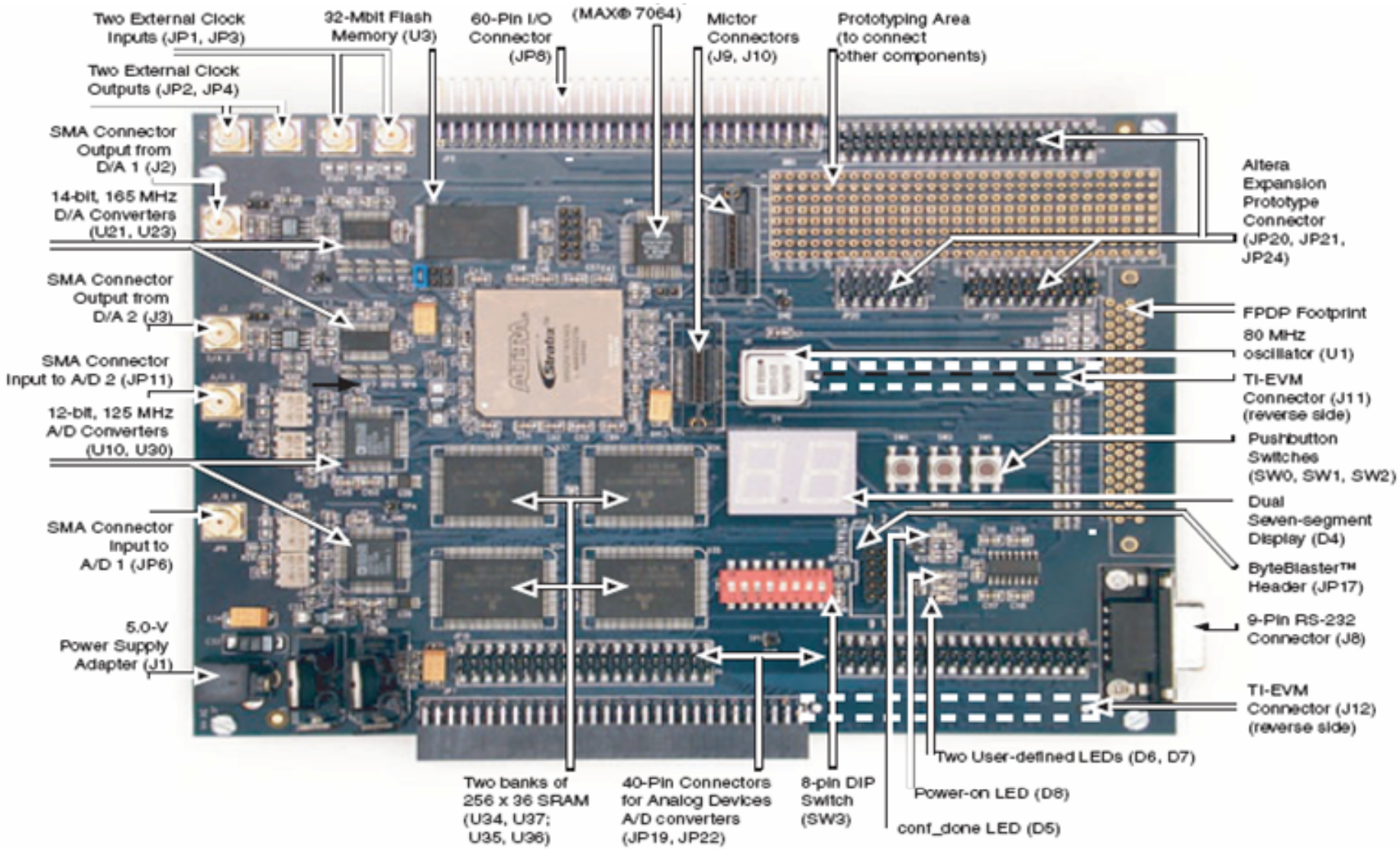


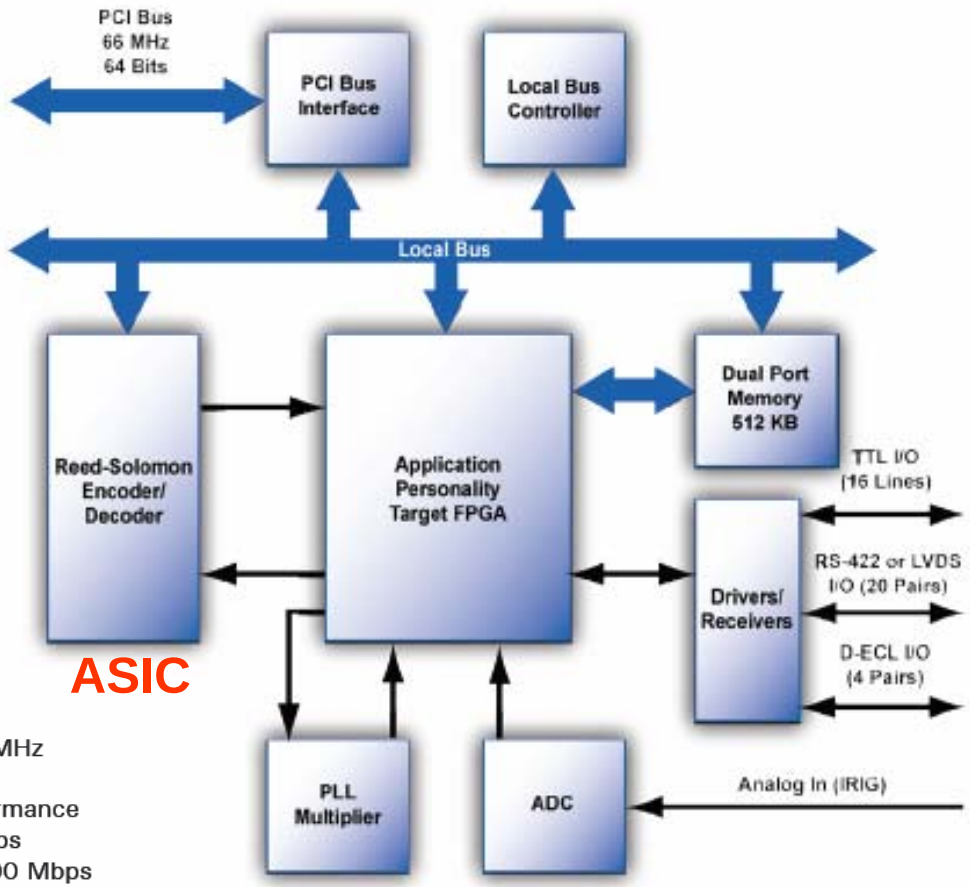
Figure 1: **Core Schematic Symbol**



Plataforma para RS (Altera)



High-Rate Front-End Processor (RT Logic)



- I/O Data Rates
  - RS-422—Up to 32 Mbps
  - LVDS—Up to 100 Mbps
  - D-ECL—Up to 700 Mbps
- Inputs/Outputs
  - Up to 16 TTL Lines.
  - Up to 20 RS-422 Pairs (40 Lines)
  - Up to 20 LVDS Pairs (40 Lines)
  - D-ECL 2 Pair Input, 2 Pair Output
  - IRIG Input (AM or DC Level Shift)
- Firmware Target
  - Up to 200 MHz Clock rate
  - Up to 8 M Logic Gates
- PCI Bus Interface
  - 528 Mbps DMA Transfer Rate
  - 66 MHz Clock rate, 64 Bits Wide
- Other Onboard Circuitry
  - 512 KB Dual-port Memory
  - PLL Multiplier, Up to 700 MHz
  - ADC for IRIG AM Input
- Application Personality Performance
  - Telemetry—Up to 450 Mbps
  - PCM Simulation—Up to 700 Mbps
  - Commanding—Up to 5 Mbps
- Physical
  - Single Wide PMC Form factor
  - Short PCI (via Carrier Board)
- Power (Typical)
  - 2A @ 5V, 3A @ 3.3 V
- Operating Environment
  - 0 to 50 °C
  - 10 to 90% Relative Humidity