Sistemas de Comunicación Digital

INF2010

Clase 2:
Introducción a los Sistemas de Comunicación II

Capacidad de un Canal

Capacidad de un Canal

- El sistema óptimo de comunicación es aquel que:
  - Minimiza la BER (Probabilidad de error de Bit) a la salida del sistema,
  - Está sujeto a las restricciones de energía transmitida y ancho de banda del canal.
- Y lo estudiamos para el caso de ruido blanco gaussiano

Capacidad de un Canal

- según Shannon,
  - Se calcula la Capacidad C de un canal [bits/s]
  - Tal que si la velocidad de información R es menor a C [bits/s]
  - Entonces la probabilidad de error será cercana a 0.

$$C = B \cdot \log_2\left(1 + \frac{S}{N}\right)$$

  - Con B como ancho de banda en Hz y S/N la relación Señal a Ruido en veces a la entrada del receptor.

$$SNR = \frac{S}{N} = \frac{P_{señal}}{P_{ruido}} = 10^{dB/10}$$

Capacidad de un Canal

- Shannon establece solo un límite, no dice cómo lograrlo.
- Podemos utilizar una medida de eficiencia η:

$$\eta = \frac{R}{C}$$

- Con R siendo el rate de señalización y C la capacidad.
- Para un R fijo, ¿si aumento el ancho de banda B, es posible reducir el SNR?
- La ecuación de Shannon nos permite comparar la eficiencia de distinto tipo de modulaciones.

Ejercicios

- Un usuario desea adquirir un modem para enviar datos por línea telefónica. Ésta tiene una SNR de 25dB y pasa frecuencias entre 300Hz y 3200Hz. Calcule la velocidad de datos que puede enviarse sobre la línea telefónica cuando no existen errores del lado receptor.

Capacidad de un Canal

En resumen

- La capacidad máxima teórica de un canal depende de:
  - Ancho de banda disponible
  - Relación Señal a Ruido (SNR)
- Y la eficiencia indica cuánto se ocupa del canal ideal con una modulación determinada.

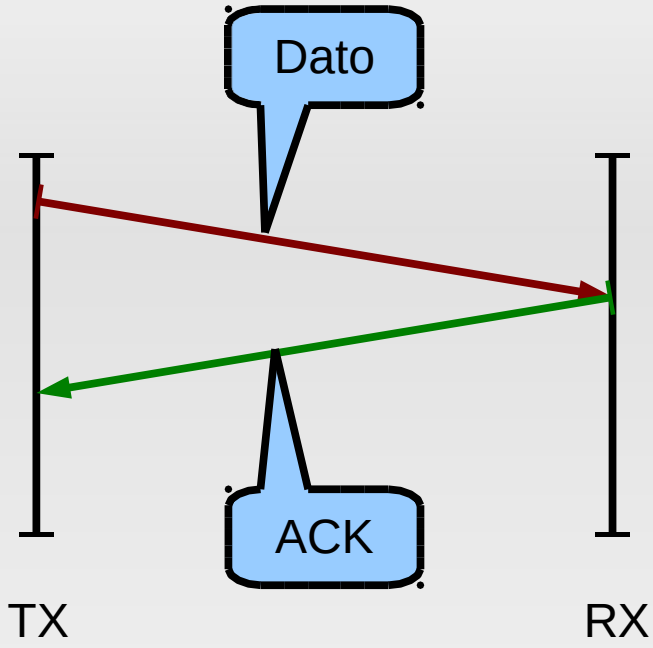
Codificación

Codificación

- Si ocurren errores en un canal, se pueden reducir usando técnicas tales como:
  - ARQ (Requisición de repetición automática)
  - FEC (Corrección de errores directa)
- En ARQ: Se detecta error, se pide retransmisión
- En FEC: Se usa para detectar y corregir errores
- Ambas son técnicas de codificación de canal (corrigen errores de canal)

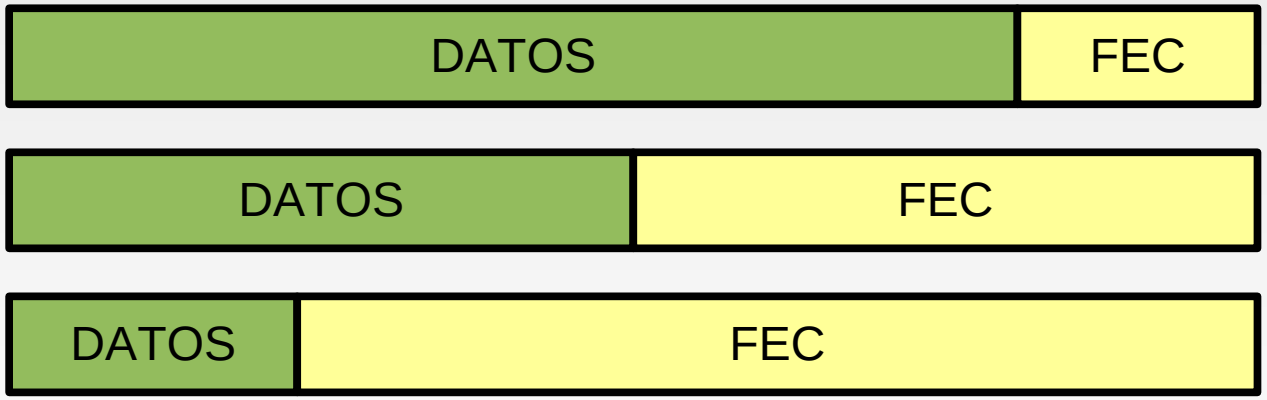
Codificación

- ARQ se usa:
  - Entre computadores,
  - De implementación de bajo costo
  - Hay canal duplex (ACK y NAC)
- FEC se usa:
  - Hay canal simplex
  - Alto delay



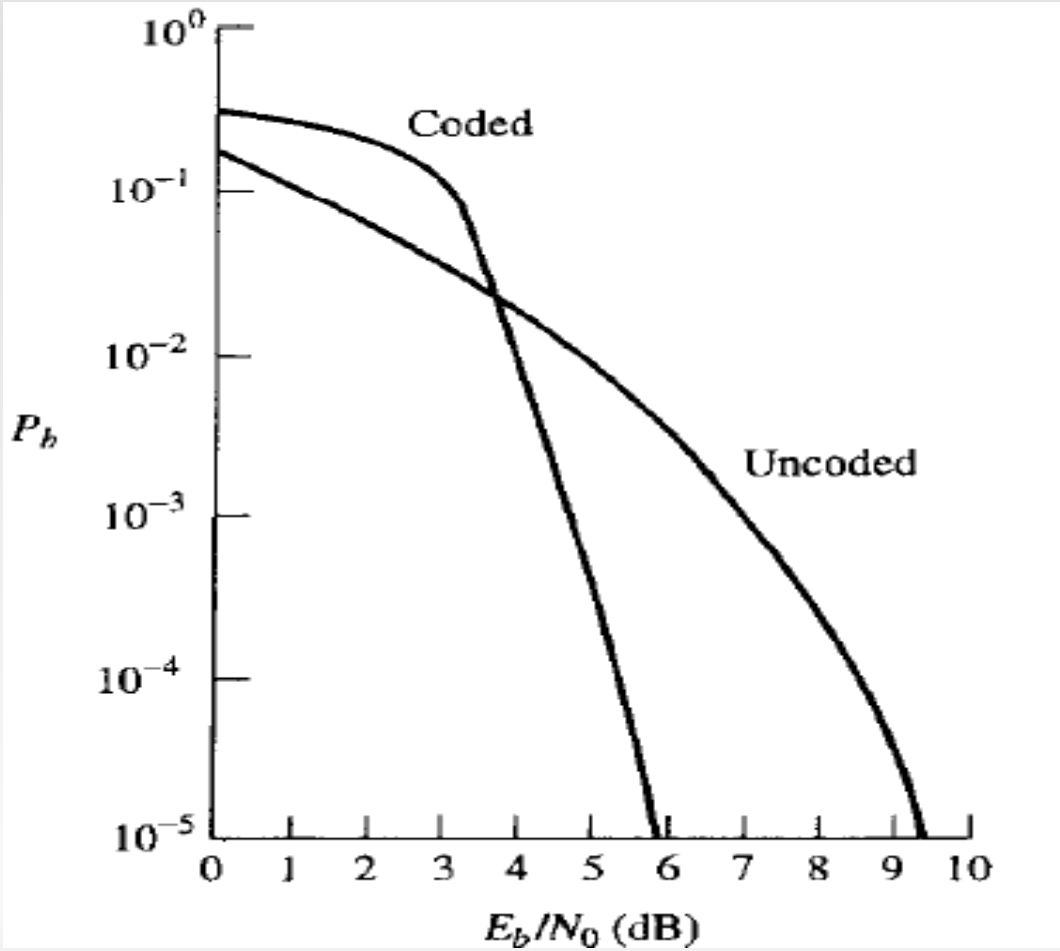
Codificación

- Nos concentramos a nivel de bit, queremos minimizar la probabilidad de error de bit: P_b
- Agregamos redundancia para que los errores sean detectables y poder corregir el máximo posible.
- Cuánta redundancia hace falta?



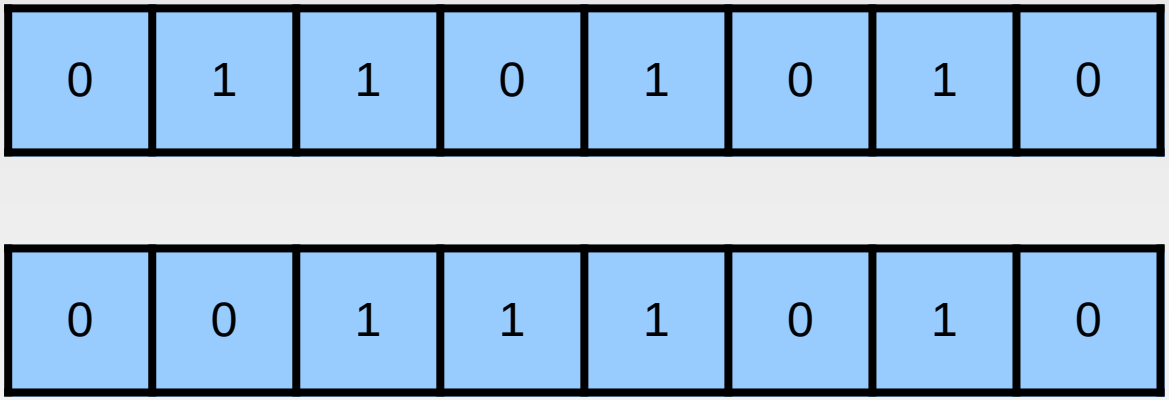
Codificación

- Cómo mejora la performance un código?



Codificación

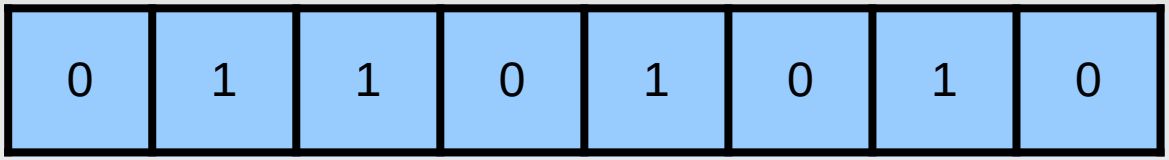
- Primer paso: Distancia de Hamming
  - Entre 2 palabras de código, en cuántos bits se diferencian.



  - Distancia=?

Codificación

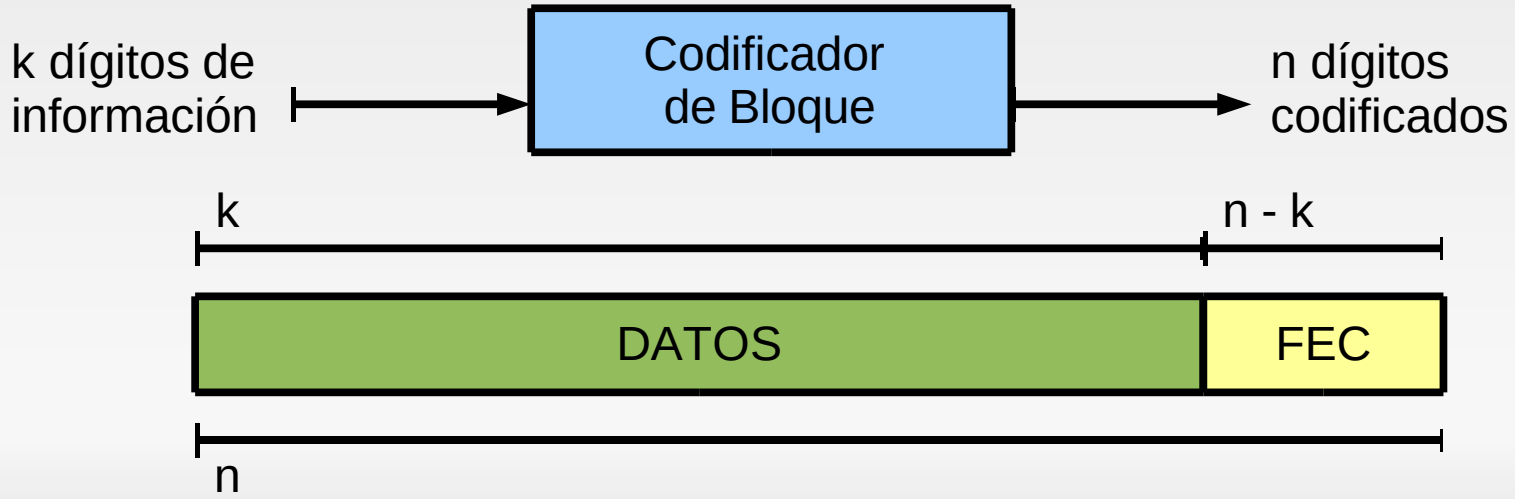
- Segundo paso: Peso
  - Es la cantidad de 1s que contiene.



  - Peso=?

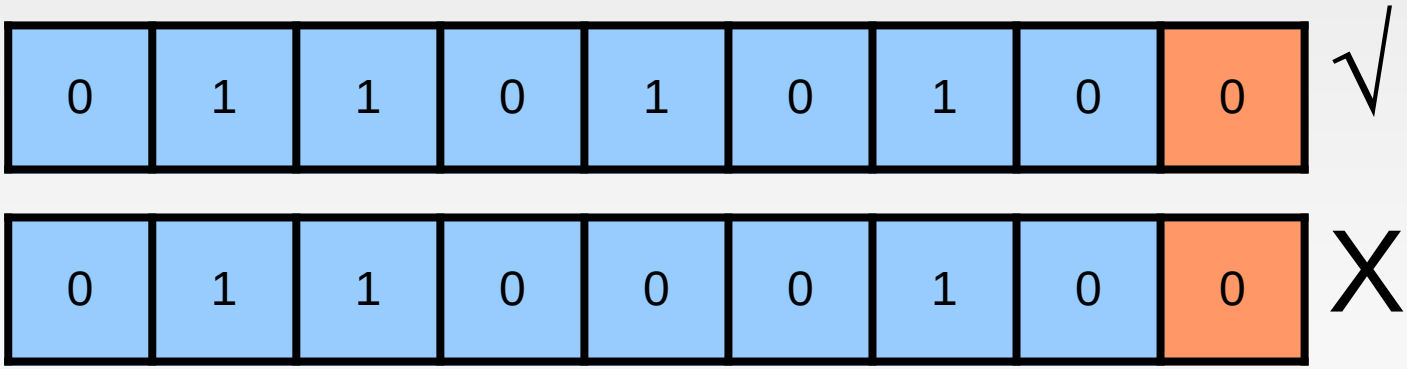
Codificación

- Tercer paso: Definición de código por bloque
  - Un código convierte una secuencia de k elementos de entrada en n elementos de salida
  - Agrega entonces (n-k) elementos redundantes.
  - La relación (Ratio) es el cociente k/n.



Codificación

- Paridad
  - La más simple redundancia es agregar un bit
  - Este bit indica si la palabra a transmitir tiene una cantidad par o impar de 1s.
  - O completa la palabra para hacer par o impar la cantidad de 1s.



Codificación de Hamming

Algoritmo:

- Posiciones: son las potencias de 2. Por ello, son bits de paridad $(2^0 = 1, 2, 4, 8, 16...)$
- El resto de las posiciones son los datos a transmitir.
- Cada bit de paridad calcula paridad de un conjunto de datos (no todos), determinados por la posición del bit de paridad. Bit 1 $(P_1 : 0001)$ calcula paridad de bits de datos con LSB a 1.

Codificación de Hamming

Ejemplo: (11, 7) p: bits de paridad, d: bits de datos
Datos = 0101001

Paso 1: Insertamos la palabra a transmitir en sus respectivas posiciones

	p1	p2	d1	p3	d2	d3	d4	p4	d5	d6	d7
Posición	1	2	3	4	5	6	7	8	9	10	11
Posición (bin)	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011
Palabra original			0		1	0	1		0	0	1
p1											
p2											
p3											
p4											
Palabra + Paridad											

Paso 2: “Bajamos” los bits de datos con LSB de posición en 1 (xxx1)

	p1	p2	d1	p3	d2	d3	d4	p4	d5	d6	d7
Posición	1	2	3	4	5	6	7	8	9	10	11
Posición (bin)	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011
Palabra original			0		1	0	1		0	0	1
p1	1		0		1		1		0		1
p2											
p3											
p4											
Palabra + Paridad											

Codificación de Hamming

Ejemplo: (11, 7) p: bits de paridad, d: bits de datos
Datos = 0101001

Paso 3: “Bajamos” los bits de datos con posición (xx1x) en 1

	p1	p2	d1	p3	d2	d3	d4	p4	d5	d6	d7
Posición	1	2	3	4	5	6	7	8	9	10	11
Posición (bin)	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011
Palabra original			0		1	0	1		0	0	1
p1	1		0		1		1		0		1
p2		0	0			0	1			0	1
p3											
p4											
Palabra + Paridad											

Paso 4: “Bajamos” los bits de datos con posición (x1xx) en 1

	p1	p2	d1	p3	d2	d3	d4	p4	d5	d6	d7
Posición	1	2	3	4	5	6	7	8	9	10	11
Posición (bin)	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011
Palabra original			0		1	0	1		0	0	1
p1	1		0		1		1		0		1
p2		0	0			0	1			0	1
p3				0	1	0	1				
p4											
Palabra + Paridad											

Codificación de Hamming

Ejemplo: (11, 7) p: bits de paridad, d: bits de datos
Datos = 0101001

Paso 5: “Bajamos” los bits de datos con posición (1xxx) en 1

	p1	p2	d1	p3	d2	d3	d4	p4	d5	d6	d7
Posición	1	2	3	4	5	6	7	8	9	10	11
Posición (bin)	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011
Palabra original			0		1	0	1		0	0	1
p1	1		0		1		1		0		1
p2		0	0			0	1			0	1
p3				0	1	0	1				
p4								1	0	0	1
Palabra + Paridad											

Paso 6: Dejamos caer todos los bits finales (cadena completa)

	p1	p2	d1	p3	d2	d3	d4	p4	d5	d6	d7
Posición	1	2	3	4	5	6	7	8	9	10	11
Posición (bin)	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011
Palabra original			0		1	0	1		0	0	1
p1	1		0		1		1		0		1
p2		0	0			0	1			0	1
p3				0	1	0	1				
p4								1	0	0	1
Palabra + Paridad	1	0	0	0	1	0	1	1	0	0	1

Codificación de Hamming

El código Hamming binario lineal cae dentro de la categoría de códigos de bloque lineales que pueden corregir errores de un solo bit.

Por cada entero p ≥ 3 (el número de bits de paridad), existe un código de Hamming (n,k).

Aquí, **n** es el número de símbolos en la palabra clave codificada (tamaño total de bloque) y **k** es el número de símbolos de información que el codificador puede aceptar a la vez.

Todos estos códigos de Hamming tienen una distancia mínima **dmin=3** y, por lo tanto, pueden corregir cualquier error de un solo bit y detectar errores de dos bits en el vector recibido.

Las características de un código de Hamming genérico (n,k) se dan a continuación.

Tamaño total del bloque	$n = 2^p - 1$
N° símbolos información	$k = 2^p - p - 1$
N° símbolos paridad	$p = n - k$
Distancia mínima	$d_{\min} = 3$
Capacidad corrección error	$t = 1$

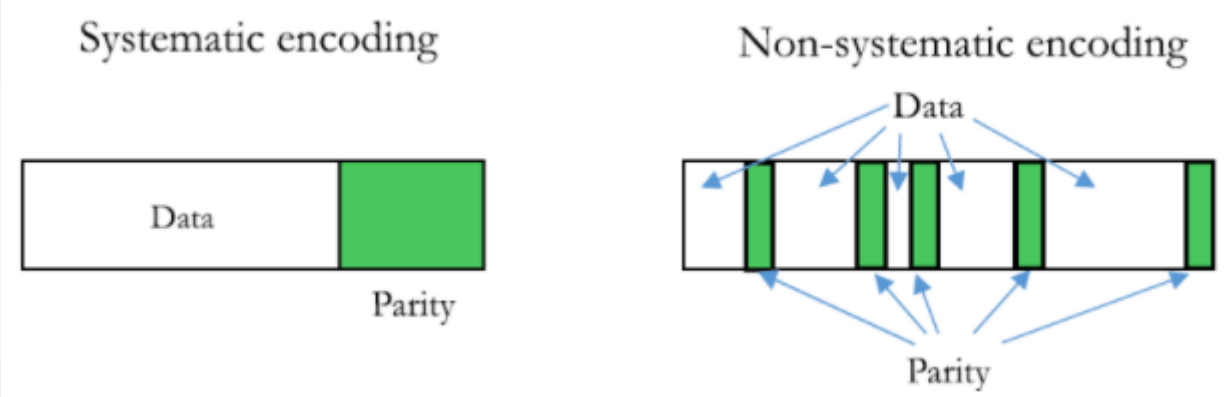
Codificación de Hamming

Codificación Sistemática y No Sistemática

Los códigos de bloque como los códigos de Hamming también se clasifican en dos categorías que difieren en cuanto a la estructura de la salida del codificador:

- Codificación sistemática
- Codificación no sistemática

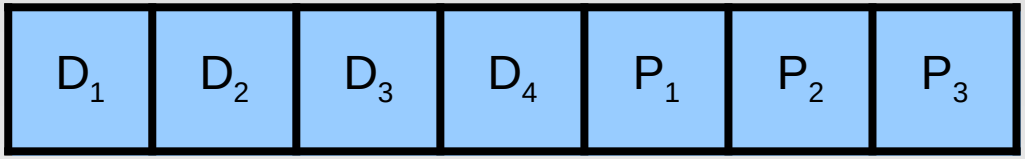
En la codificación sistemática, con solo ver la salida de un codificador, podemos separar los datos y los bits redundantes (también llamados bits de paridad). En la codificación no sistemática, los bits redundantes y los bits de datos se intercalan.



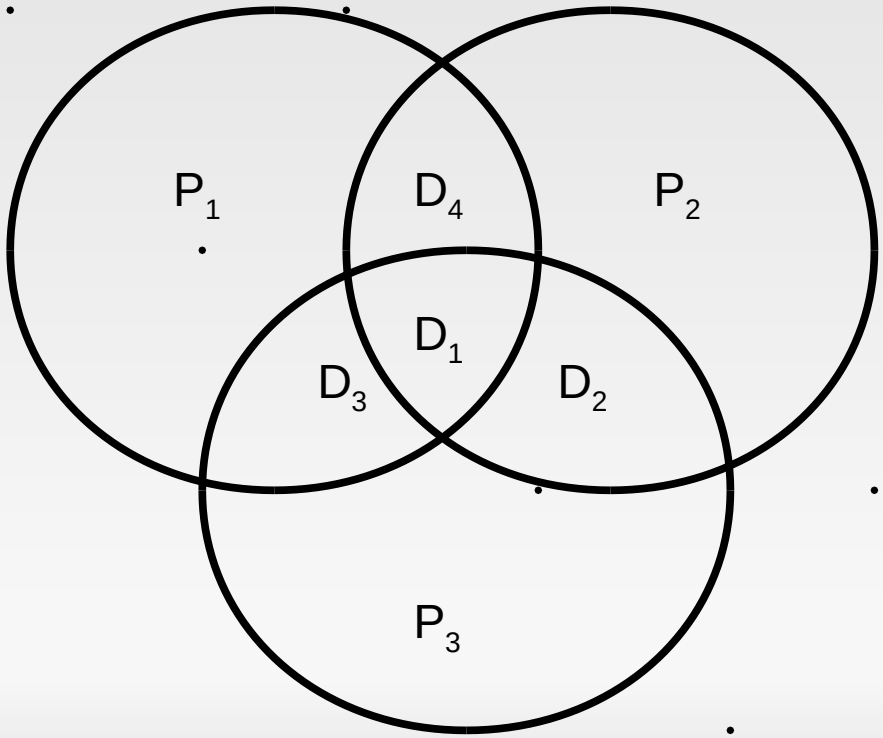
Codificación

La figura ilustra una palabra de código de 7 bits con 4 dígitos de información (D1 a D4) y 3 dígitos de chequeo de paridad (P1 a P3), que es un bloque (7,4). Suponiendo paridad PAR, muestre la realización de este codificador usando sumadores módulo-2 de tres entradas. Calcule los bits individuales de chequeo de paridad y codificación de P1, P2 y P3, para los bits de información 1011

- Ejemplo de codificador

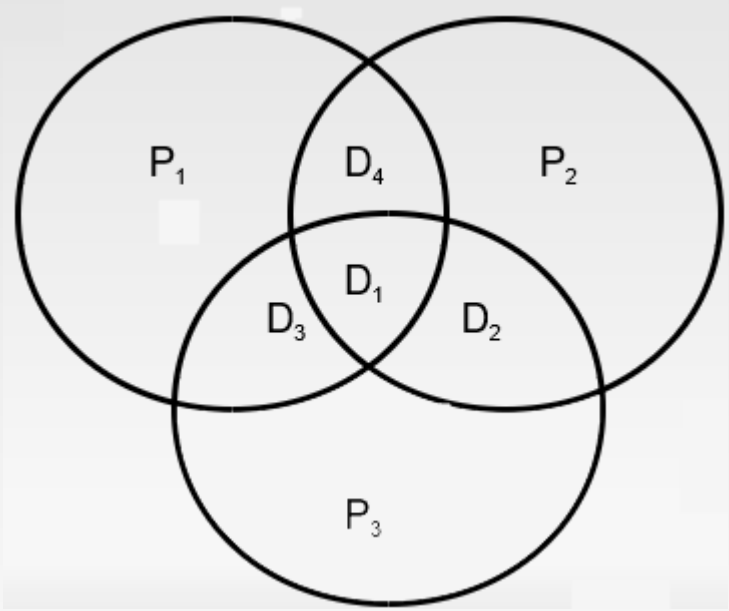


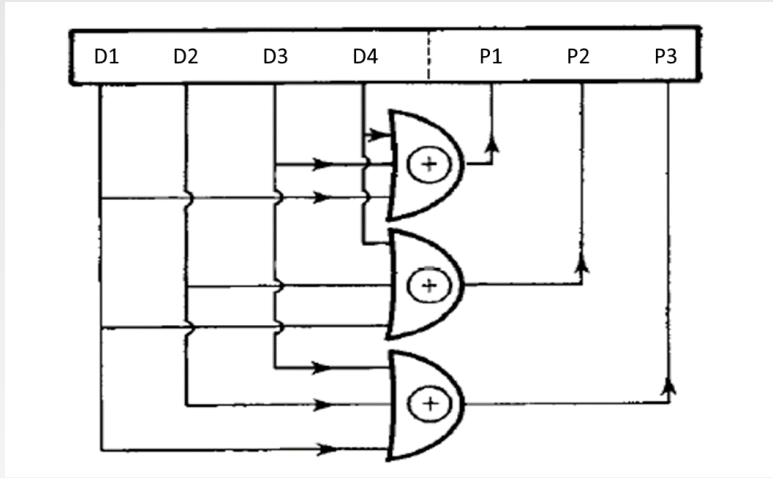
- n=7, k=4, n-k=3



Codificación

- Y su circuito, con compuertas or exclusivo de multiples entradas:





Codificación

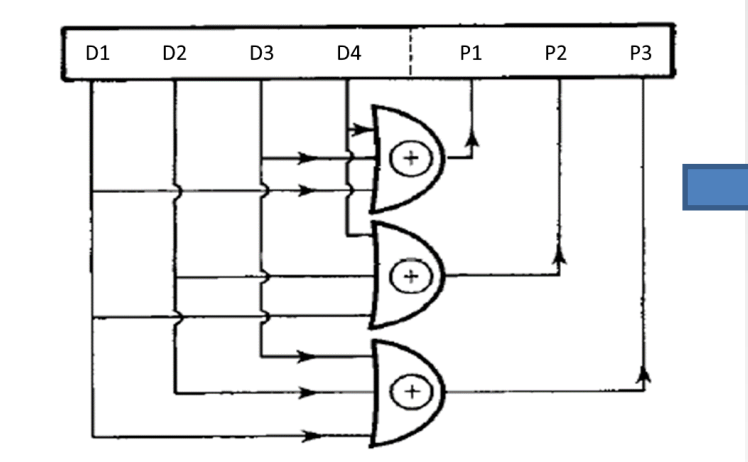
- Matriz Generadora [**k** x **n**]:

En este caso:
n = 7 , k = 4, p = 3

$$[G] = [I_k : P]$$

$[P]$: Matriz [k × p] que contiene coeficientes de paridad

$[I_k]$: Matriz identidad [k × k]



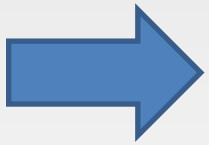
$$P_1 = 1 \times D_1 \oplus 0 \times D_2 \oplus 1 \times D_3 \oplus 1 \times D_4$$
$$P_2 = 1 \times D_1 \oplus 1 \times D_2 \oplus 0 \times D_3 \oplus 1 \times D_4$$
$$P_3 = 1 \times D_1 \oplus 1 \times D_2 \oplus 1 \times D_3 \oplus 0 \times D_4$$

$$G = \left[\begin{array}{cccc:ccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{array}\right]$$

Codificación

- Matriz Generadora (7,4):

$$G = \left[\begin{array}{cccc:ccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{array}\right]$$



Palabra codificada			
Informacion	P1	P2	P3
0000	0	0	0
0001	1	1	0
0010	1	0	1
0011	0	1	1
0100	0	1	1
0101	1	0	1
0110	1	1	0
0111	1	1	1
1000	1	1	1
1001	0	0	1
1010	0	1	0
1011	1	0	0
1100	1	0	0
1101	0	1	0
1110	0	0	1
1111	1	1	1

Codificación

- Matriz de Corrección de error (H). $[H] = [P^T \vdots I_p]$

$$P_1 = 1 \times D_1 \oplus 0 \times D_2 \oplus 1 \times D_3 \oplus 1 \times D_4$$
$$P_2 = 1 \times D_1 \oplus 1 \times D_2 \oplus 0 \times D_3 \oplus 1 \times D_4$$
$$P_3 = 1 \times D_1 \oplus 1 \times D_2 \oplus 1 \times D_3 \oplus 0 \times D_4$$

$$H = \left[\begin{array}{cccc:ccc} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

Codificación

- Probabilidad de error:
- Supongamos P_e = probabilidad de error de 1 bit
- R' = cantidad de errores, n = longitud de bloque
- Buscamos la probabilidad de tener más de R' errores en un bloque de n dígitos.

$$P(e > R' \, errores) = 1 - P(e < R' \, errores)$$

$$P(e > R' \, errores) = 1 - [P(0 \, errores) + P(1 \, error) + P(2 \, errores) + \cdots + P(R' \, errores)]$$

- Y las probabilidades se calculan individualmente

Codificación

- Un bloque que representa una palabra de código se divide en elementos desde 1 a n.
- Cada elemento corresponde a un dígito en una palabra de n dígitos, y se le asigna la probabilidad que le corresponde.
- Si consideramos el caso general, j errores en n dígitos nos da (binomial):

$$P(j\_errores) = \left(P_e\right)^j \left(1-P_e\right)^{n-j} \cdot {}^nC_j$$

$${}^nC_j = \frac{n!}{j!(n-j)!} = \binom{n}{j}$$

Codificación

- Volviendo a la ecuación:

$$P(e > R' \, errores) = 1 - [P(0 \, errores) + P(1 \, error) + P(2 \, errores) + \cdots + P(R' \, errores)]$$

- Podemos reescribirla:

$$P(e > R' \, errores) = 1 - \sum_{j=0}^{R'} P(j \, errores)$$

- siendo:

$$P(j \, errores) = (P_e)^j (1 - P_e)^{n-j} \cdot {}^{n}C_j$$

- Entonces, si hacemos n muy grande, la cantidad de errores en un bloque va a tender a $P_e n$.

Codificación

Ejemplo:

- La probabilidad de un error simple es 0.01, entonces la probabilidad de recibirlo correctamente es 0.99.
- Calcular la probabilidad de 0 a 2 errores que ocurran en una palabra de 10 dígitos.

Codificación

- Siendo todas las recepciones de bit independientes, la probabilidad de no tener errores en el bloque es:

$$(0.99)^{10}=0.904382$$

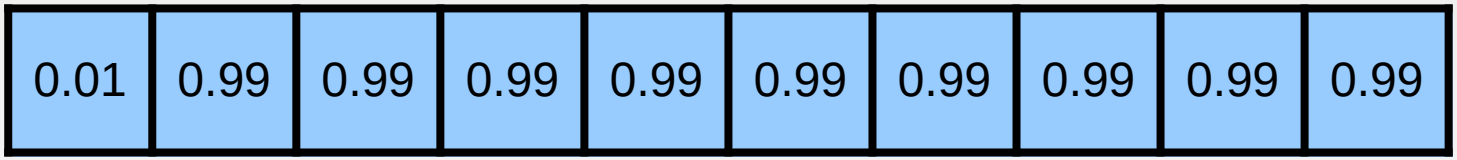
0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99

Codificación

- Probemos ahora con 1 solo error. El error está en el primer lugar. La probabilidad es 0.01 y la probabilidad de recibir al resto bien es de 0.99 cada uno (y hay 9):

$$P(1\,error) = (0.01)^1 (0.99)^9 \cdot {}^{10}C_1 = 0.091352$$

- ${}^{10}C_1$ = 10 es el número de combinaciones de 1 objeto de 10 objetos.



Codificación

- Para 2 errores:

$$P(2\,errores)=(0.01)^2(0.99)^{8}\cdot{}^{10}C_2=0.00415$$

- Entonces la probabilidad de 3 o más errores es:

$$1-0.904382-0.091352-0.00415=0.000116$$

- La probabilidad de j errores en el bloque se reduce rápidamente con j.

Codificación

- Entonces, si elegimos un código que pueda corregir $P_e n$ errores en un bloque,
- Van a existir pocos casos en que la codificación falle.
- Los más interesantes entre los códigos de bloque son los códigos lineales.

Codificación

Códigos Lineales de Grupo

- Contienen la palabra con todos ceros
- Si se toman dos palabras C_i y C_j entonces:

$$C_i \oplus C_j = C_k$$

- La operación de suma módulo-2 de 2 palabras da otra palabra del mismo código.

Codificación

- Veamos un ejemplo (alfabeto de 4 miembros, a,b,c,d)
- Se codifican en palabras de 5 dígitos (n=5)
- El código es (5,2)
- Si sumamos las palabras de c y b, obtenemos d.

$$\begin{array}{ll} a = & 0\ 0 \\ b = & 0\ 1 \\ c = & 1\ 0 \\ d = & \underbrace{1\ 1}_{k=2} \end{array} \qquad \begin{array}{c} 0\ 0\ 0\ 0\ 0 \\ 0\ 0\ 1\ 1\ 1 \\ 1\ 1\ 1\ 0\ 0 \\ \underbrace{1\ 1\ 0\ 1\ 1}_{n=5} \end{array}$$

$$\begin{array}{lll} c \oplus b = & d & \\ c = & 1\ 1\ 1\ 0\ 0 & \\ b = & 0\ 0\ 1\ 1\ 1 & \\ \hline d = & 1\ 1\ 0\ 1\ 1 & \end{array}$$

Codificación

- Los más usados son los generados por polinomios.
- Para implementarlos, se utilizan Shift Registers.
- De los polinómicos, los más comunes son los BCH (Bose-Chaudhuri-Hocquenghem) y los Reed-Solomon.
- Los polinomios BCH están tabulados hasta n=255 y pueden corregir hasta 30 dígitos
- Los Reed-Solomon (no binarios) son utilizados en los Cds.

Codificación

Performance de un código

- Se mide considerando todas las distancias de Hamming entre pares de códigos
- Realmente hace falta medir solo la distancia respecto de la palabra con todos ceros.
- La que cuenta es la distancia mínima.
- En el ejemplo, la D_{min} es?
- Para este código (5,2)
- Como la distancia es respecto de 0, entonces corresponde al peso de cada palabra

0	0	0	0	0
0	0	1	1	1
1	1	1	0	0
1	1	0	1	1

Codificación

- Supongamos que una palabra C_i es confundida con una palabra C_j.
- La probabilidad depende de la distancia entre ambas: D_{ij}.
- La distancia equivale a una palabra C_k, suma módulo-2 de C_i y C_j.
- La probabilidad de confundir una con la otra es la misma que confundirse entre C_k y C_0.
- Y depende solo del peso de C_k.

Codificación

Capacidad de corrección de errores:

- Definimos t como la máxima posibilidad de corregir todos los patrones de error de t o menos errores.
- Respecto a la distancia de Hamming:

$$t = \text{int}\left(\frac{D_{\min} - 1}{2}\right)$$

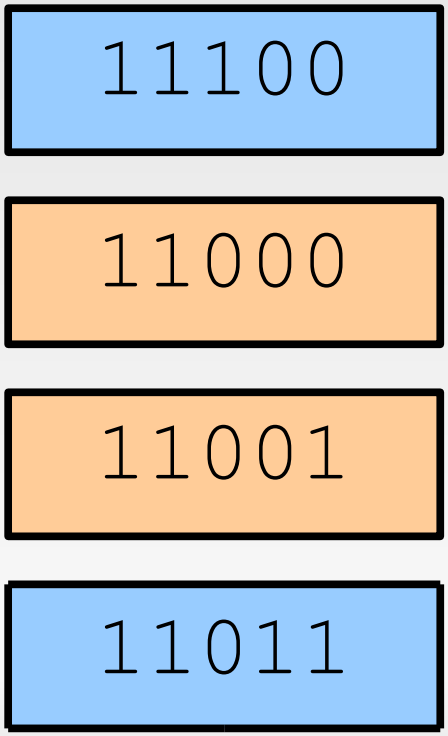
- Donde

$$D_{min} - 1 = e + t$$

- Int es parte entera, e es la cantidad de errores detectables (incluyendo los t corregibles) y $t \leq e$

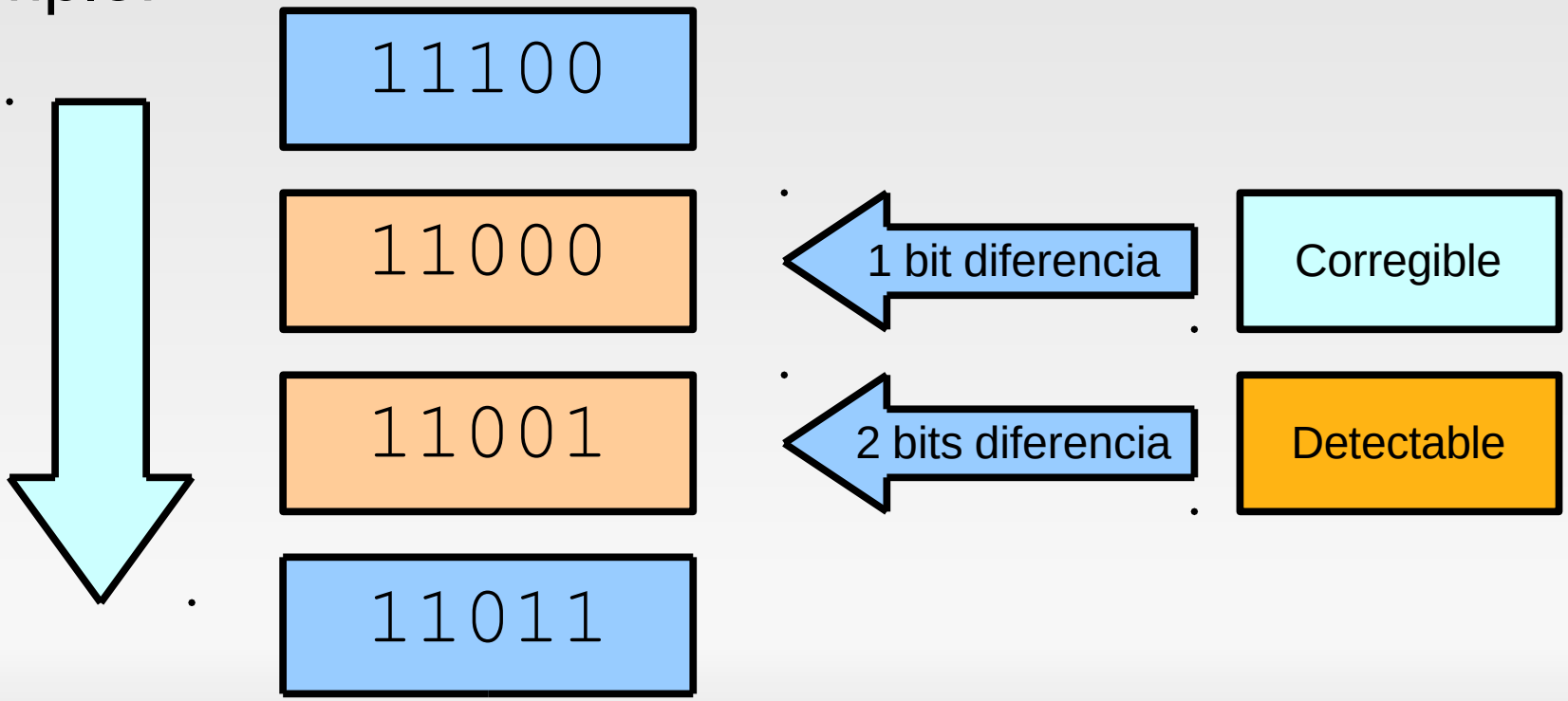
Codificación

- Supongamos que tenemos un código con $D_{min}=3$.
- Hay entonces 2 palabras binarias entre las dos palabras válidas.
- Ejemplo:



Codificación

- Si ocurre 1 solo error de bit, este puede ser corregido.
- En cambio, si no hay corrección, se pueden detectar dos errores.
- Ejemplo:



Codificación

- Si los códigos son más largos, entonces podemos detectar y corregir más bits.
- Para un D_{min}=7, t=1 y e=5, o t=2 y e=4.
- Considerando la eficiencia, un código BCH con n=63 y k=57, R=0,9 con t=1 bit.
- Si reducimos k a 45, R=0,7 pero t=3 bits.

Codificación

Cómo decodificamos?

- Hay 2 estrategias:
  - El vecino más cercano
  - Máximo parecido (Maximmum Likelihood)
- La capacidad es parecida si la probabilidad de t errores es mucho mayor que la de t+1 errores.
- Además, se denomina código perfecto a aquel que elimina las ambiguedades.

Codificación

Ejemplo:

- Tomemos la tabla:

$$H = \left[\begin{array}{cccc:ccc} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

Palabras de código				
	00000	11100	00111	11011
Corregible 1 bit	10000	01100	10111	01011
	01000	10100	01111	10011
	00100	11000	00011	11111
	00010	11110	00101	11001
	00001	11101	00110	11010
Detectable 2 bits	10001	01101	10110	01010
	10010	01110	10101	01001

Codificación

Decodificación por síndrome

- El método de buscar el vecino más cercano o el más parecido se complica si el código crece.
- Proponemos una solución con la matriz de generación.
- Usaremos 2 matrices: la H (de paridad) y la G (de generación)

Codificación

Ejemplo:

- Para un código (7,4)
- La matriz de generación arma un código de largo n
- A partir de una secuencia de k dígitos.

$$\mathbf{H} = \left[\begin{array}{ccccccc} 1 & 0 & 1 & 1 & : & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & : & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & : & 0 & 0 & 1 \end{array}\right]$$

$$\mathbf{G} = \left[\begin{array}{ccccccc} 1 & 0 & 0 & 0 & : & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & : & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & : & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & : & 1 & 1 & 0 \end{array}\right]$$

Codificación

- Lo que está a la derecha de G
- Es lo que está a la izquierda de H, transpuesto.
- Como podemos corregir 1 solo bit, D_{min} y el peso deben ser 3.

$$\mathbf{H} = \left[\begin{array}{cccc:ccc} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

$$\mathbf{G} = \left[\begin{array}{cccc:ccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{array}\right]$$

Codificación

- Además, por regla, el costado derecho debe tener al menos dos 1s,
- Y no pueden haber 2 filas idénticas.

$$\mathbf{H} = \left[\begin{array}{cccc:ccc} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

$$\mathbf{G} = \left[\begin{array}{cccc:ccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{array}\right]$$

Codificación

- Además, por regla, el costado derecho debe tener al menos dos 1s,
- Y no pueden haber 2 filas idénticas.

$$\mathbf{H} = \left[\begin{array}{cccc:ccc} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

$$\mathbf{G} = \left[\begin{array}{cccc:ccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{array}\right]$$

Codificación

¿Cómo se codifica?

- Supongamos un dato 1001.

$$[1\ 0\ 0\ 1]\begin{bmatrix} 1\ 0\ 0\ 0\ 1\ 1\ 1 \\ 0\ 1\ 0\ 0\ 0\ 1\ 1 \\ 0\ 0\ 1\ 0\ 1\ 0\ 1 \\ 0\ 0\ 0\ 1\ 1\ 1\ 0 \end{bmatrix} = [1\ 0\ 0\ 1\ 0\ 0\ 1]$$

- Aparecen el dato y sus bits de paridad correspondientes.

Codificación

¿Cómo se decodifica?

- Creamos una tabla de síndrome.
- El síndrome no depende de la cantidad de códigos sino de la secuencia de errores.
- Sabemos que **d.G=c**, y que **Hc=0.**
- Propongamos **r**, secuencia recibida cuando se transmite **c**,
- y **e** es el vector de ubicación de errores en la recepción:

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

Codificación

- Creamos el producto de **H** y **r**:

$$\mathbf{s} = \mathbf{H}\,\mathbf{r} = \mathbf{H}\,(\mathbf{c} \oplus \mathbf{e})$$

$$= \mathbf{H}\,\mathbf{c} \oplus \mathbf{H}\,\mathbf{e} = \mathbf{0} \oplus \mathbf{H}\,\mathbf{e}$$

- Vemos entonces que si no hay errores, el sindrome va a ser **0**.
- Más aún, calculando **s**, obtenemos **e**, que es la posición de los errores.

Codificación

- Una tabla de síndrome se construye suponiendo que se transmite la palabra nula:

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

Codificación

- Y luego buscando todos los patrones de 1 bit de error posibles asociados a esa palabra.

Patrón de error	Síndrome
0 0 0 0 0 0 0	0 0 0
1 0 0 0 0 0 0	1 1 1
0 1 0 0 0 0 0	0 1 1
0 0 1 0 0 0 0	1 0 1
0 0 0 1 0 0 0	1 1 0
0 0 0 0 1 0 0	1 0 0
0 0 0 0 0 1 0	0 1 0
0 0 0 0 0 0 1	0 0 1

Porque en este caso solo se pueden corregir errores de 1 bit.

Codificación

En resumen

- Un código agrega redundancia para detectar y/o corregir errores en una secuencia de bits.
- La distancia de Hamming mide el grado de similaridad entre dos secuencias de bit del mismo largo
- La matriz de corrección de error es generada a partir de las ecuaciones que corresponden a la implementación del código.

Codificación II

- Códigos Cíclicos
- Entrelazado

Codificación II

Ejemplo de decodificación por síndrome

- Supongamos que recibimos para el código (7,4)

  r= `1001101`

- Encuentre el valor transmitido

Codificación II

- Encontramos el síndrome para esa palabra

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

Codificación II

- Buscamos `100` en la tabla de síndromes

Patrón de error	Síndrome
0 0 0 0 0 0 0	0 0 0
1 0 0 0 0 0 0	1 1 1
0 1 0 0 0 0 0	0 1 1
0 0 1 0 0 0 0	1 0 1
0 0 0 1 0 0 0	1 1 0
0 0 0 0 1 0 0	1 0 0
0 0 0 0 0 1 0	0 1 0
0 0 0 0 0 0 1	0 0 1

- Vemos que corresponde a

`0000100`

Codificación II

- Finalmente,

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

- Entonces:

r= 1001101

e= 0000100

---

c= 1001001

Codificación II

Ejercicio:

- Para un código lineal por bloque (6,3), la palabra tiene I_1, I_2, I_3, P_1, P_2 y P_3.
- Los bits de paridad P_1, P_2 y P_3 se arman:

$$P_1 = I_1 \oplus I_2$$
$$P_2 = I_1 \oplus I_3$$
$$P_3 = I_2 \oplus I_3$$

- Encuentre la matriz de paridad, la matriz de generación, todas las palabras posibles, determine el

Codificación II

Encuentre:

- a) la matriz de paridad,
- b) la matriz de generación,
- c) todas las palabras posibles,

Determine

- d) el peso mínimo
- e) la mínima distancia
- f) la capacidad de detección y corrección de error de este código
- g) Si la secuencia recibida es 101000, calcule el síndrome y decodifique la secuencia recibida.

Codificación II

Solución:

- a) Matriz de Paridad

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

- b) Matriz de Generación

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

Codificación II

Solución:

- c) y d) Códigos y pesos:

Mensaje x G =	Palabra	Peso
000 x G=	000000	0
001 x G=	001011	3
010 x G=	010101	3
100 x G=	100110	3
011 x G=	011110	4
101 x G=	101101	4
110 x G=	110011	4
111 x G=	111000	3

Codificación II

Solución:

- e) D_{min}= Peso mínimo = 3
- f) Entonces el código permite corregir 1 error o detectar 2 errores.
- g) Como **H.r =s**,

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

Codificación II

Solución:

- La palabra podría ser decodificada si construimos síndromes para todos los patrones de error posibles
- Y la palabra decodificada es:

111000

- Siendo esta palabra más cercana al patrón de bits recibido.

Codificación II

Códigos Cíclicos

- Estos son códigos de grupo que no contienen la palabra nula (todos 0).
- Por ejemplo, el código de Hamming.
- Permite códigos de corrección de mayor orden
- Se pueden armar con registros de corrimiento y con compuertas XOR
- Las palabras son sólo corrimientos de otras palabras
- Se pueden representar como polinomios

Codificación II

- El corrimiento se puede escribir como:

$$C = (I_{1,} I_{2,} I_{3,} \cdots, I_n)$$

- Entonces:

$$C_i = (I_{i+1}, \cdots, I_n, I_{1,} I_{2,} I_{3,} \cdots, I_i)$$

- C_i es un corrimiento de i-ésimo bit de C, e $I_1,...,I_n$ representan tanto información como paridad.
- Por ejemplo:

1001011
0101110
0010111

Codificación II

- La codificación se realiza multiplicando el vector de datos por un polinomio generador.
- Un ejemplo es el código Reed-Solomon.
- Pero éste trabaja sobre símbolos en lugar de bits individuales.
- En general, son n símbolos donde cada símbolo tiene m bits de largo.
- Un símbolo, si está compuesto por m=8 bits, es un byte.

Codificación II

- Los códigos Reed-Solomon son buenos corrigiendo ráfagas (bursts) de errores.
- Pueden corregir: $t = \frac{n-k}{2}$
- Donde n y k son símbolos, no bits.

Ejemplo: Reed-Solomon (31,15) tiene 31 códigos de 5 bits de entrada,

- Que representan 15 símbolos de información de entrada o 75 bits de información.
- Puede corregir 8 errores de bit independientes o 4 ráfagas de longitud igual o menor a los 5 bits del símbolo

Codificación II

CRC (Códigos de Redundancia Cíclica)

- Se utilizan para detectar (más que corregir) errores en canales seriales
- Utilizan aritmética donde la suma y la resta son módulo2 (sin carry), osea XOR.
- Un mensaje de k bits: $m_{k-1},\ldots,m_1,m_0$ se podría representar como un polinomio de orden k-1:

$$M(x) = m_{k-1} x^{k-1} + \cdots + m_1 x + m_0$$

Codificación II

- M(x) entonces se modifica con el polinomio generador P(x) y se forma la versión codificada de M(x).
- Se logra multiplicando (corriendo los bits hacia la izquierda) a M(x) por el orden de P(x).
- P(x) se divide entre la versión de M(x) con corrimiento y el resto es agregado al final de M(x) reemplazando los ceros a la derecha provenientes del corrimiento.
- Se descarta el resultado del cociente.

Codificación II

Ejemplo:

- Genere una palabra de un código polinomial de la secuencia de datos 1001 y el polinomio:

$$1+ x+ x^3$$

- Este se denomina código de Hamming (7,4).

Codificación II

Solución:

- M(x)=1001, equivalente a $1+x^3$ y el polinomio P(x)=1101 , equivalente a $1+x+x^3$.
- M(x) con corrimiento del orden de P(x) es: 1001000.
- Se divide por P(x) y el resto es 011.
- Se reemplazan los 3 ceros agregados y da 1001011.

```
          1111
1101 ⌈1001000
     −1101
      100000
     −1101
       10100
      −1101
        1110
       −1101
         011
```

Codificación II

Entrelazado

- Para distribuir los errores en ráfaga (y ayudar a los detectores y correctores)
- Se altera el orden de los bits a transmitir.

Codificación II

- Por ejemplo, se escribe por columna y se lee por fila: