SIMULACION EN ORDENADOR DE UNA CADENA COMPLETA DE TRANSMISION DE MENSAJES BINARIOS EN CODIGOS BCH, CON DECODIFICACION PARA CORRECCION DE ERRORES MULTIPLES

Por F. SAEZ VACAS
Profesor encargado de la cátedra de Ordenadores.
E. T. S. I. Telecomunicación. Honeywell Bull, S. A.

J. M. HERNANDO
Profesor encargado de la cátedra de Sistemas de Telecomunicación.
E. T. S. I. Telecomunicación.

B. FONTANA
I. T. T. Laboratorios de España.

(Texto íntegro de la comunicación presentada en el Congreso de Automática de Madrid, 14 de abril de 1970.)

1. INTRODUCCION

La figura 1.1 representa el esquema típico de una comunicación digital binaria, en el que suponemos mensajes constituidos por bloques de bits de una determinada longitud.

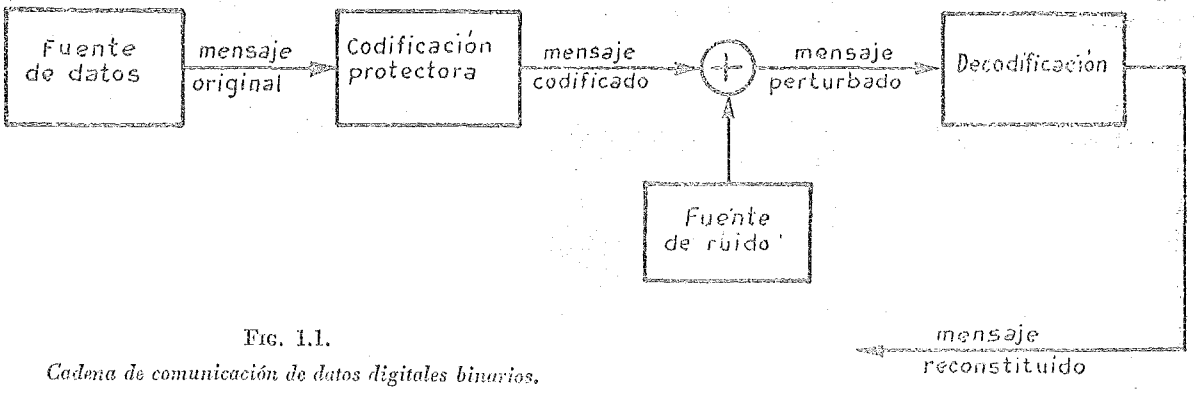

FIG. 1.1.
Cadena de comunicación de datos digitales binarios.

La finalidad última de codificación y decodificación es conseguir que el mensaje reconstituido sea idéntico al original.

En la teoría de códigos los mensajes binarios se caracterizan por vectores o también por polinomios con coeficientes pertenecientes al cuerpo de Galois GF (2), $\{0,1\}$. Sobre los conceptos de código, código lineal, código cíclico, generación polinómica de códigos, distancia, síndrome, relaciones con los elementos de un cuerpo finito, detección y corrección, etc., el mejor autor de referencia sigue siendo Peterson [1].

La función de un decodificador corrector es establecer una correspondencia entre el síndrome del mensaje perturbado y un vector ruido, o vector de error. Sustrayendo del vector recibido dicho vector de error, se obtiene el vector o mensaje codificado que, en el caso de códigos separables, contiene el mensaje original, lo que viene expresado por la figura 1.2. La idea y el fundamento teórico son conocidos desde hace años (consúltese Peterson [1]), pero su realización instrumental es tanto más complicada cuanto más ambicioso es el código. En cuanto a los códigos BCH, sucesivamente han ofrecido soluciones Peterson, Zieler, Meggitt, Bartee y Schneider, estos dos últimos realizando un autén-

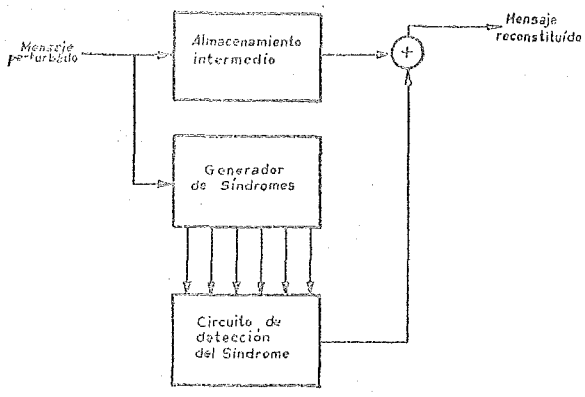

FIG. 1.2.

Decodificador

tico computador de propósito especial que decodificaba códigos BCH correctores de hasta cinco errores. Sin embargo, sólo recientemente se han publicado procedimientos que simplifican respectivamente alguna de las fases de la decodificación. Se deben a Chien [2] y Berlekamp [3] y [4].

Siguiendo a este último, el esquema de la figura 1.2 puede ampliarse y precisarse algo más en el esquema de la figura 1.3, donde se aprecia que la detección de los síndromes y localización de errores incluye operaciones en un cuerpo de Galois.

Los códigos BCH, recordémoslo, constituyen una clase de códigos correctores de errores múltiples que pueden describirse en función de los polinomios mínimos $m_i(x)$ de la forma siguiente:

Sea $g(x) = mcm\,\{m_1(x) : m_3(x) : \ldots m_{2t-1}(x)\}$ el polinomio generador. El código generado por $g(x)$ es un código que corrige t errores, con una distancia mínima de por lo menos $2t + 1$ y una longitud $n = e_1$, siendo e_1 el período de $m_1(x)$.

Otra forma equivalente de describirlos es la siguiente:

Si α es un elemento primitivo del cuerpo finito GF(2^m), el código BCH corrector de t errores puede definirse como el conjunto de todos los polinomios $\{b(x)\}$ sobre GF(2), de grado $n - 1$ o menos, tales que

$$b(\alpha^i) = 0, \quad i = 1, 3, 5, \ldots (2t - 1)$$

donde

$$b(x) = b_0 + b_1x + b_2x^2 + \ldots + b_{n-1}x^{n-1} \quad \text{y} \quad b_i = 0,1.$$

Un vector o mensaje codificado, generado con arreglo a esta definición, tiene mt bits de control.

Los síndromes (S) son los restos resultantes

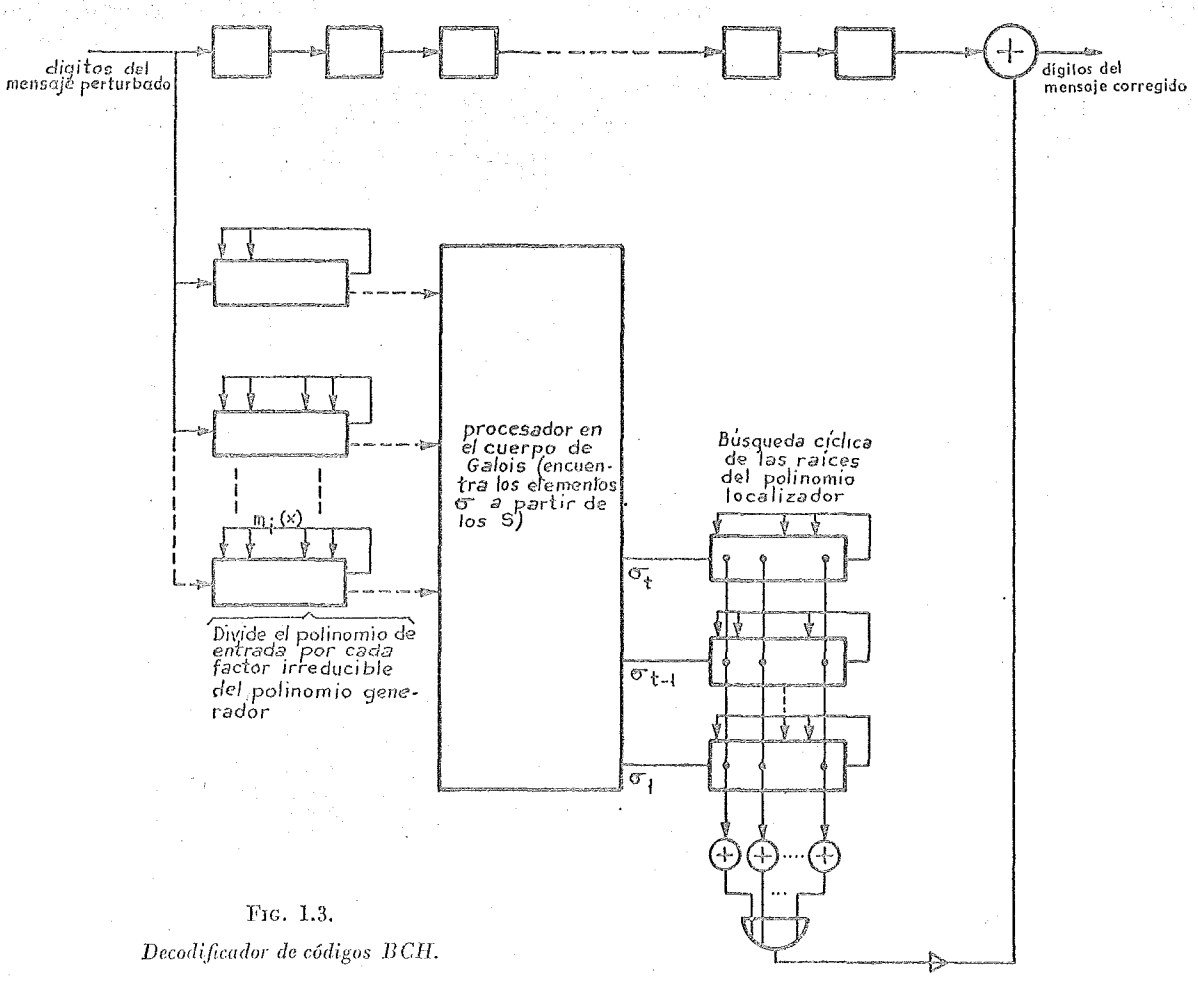

FIG. 1.3.

Decodificador de códigos BCH.

de dividir el polinomio correspondiente al mensaje perturbado, por cada uno de los polinomios $m_i(x)$, dichos restos valorados para diferentes potencias de α. Son, por tanto, elementos del cuerpo GF(2^m).

Berlekamp [4] ha descrito un método para calcular los coeficientes σ_i del polinomio localizador de errores: $\sigma(x)$ a partir de los elementos S, y Chien, anteriormente [2], un método muy original que corrige los errores, a partir de los coeficientes σ_i, sin necesidad de calcular las raíces de $\sigma(x)$. Este procedimiento aprovecha de manera magistral y simultáneamente las propiedades algebraicas y cíclicas de los códigos.

La instrumentación física de los diferentes órganos se realiza a base de circuitos secuenciales lineales constituidos por conexiones adecuadas de los siguientes componentes (fig. 1.4):

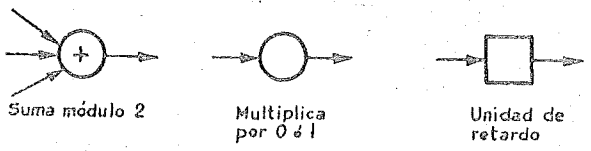

FIG. 1.4.

Componentes primarios de los circuitos secuenciales lineales.

así como de circuitos combinatorios. (Consúltese cualquiera de las referencias [1] a [4] y, si se quiere profundizar en este tipo de circuitos, independientemente del estudio de códigos, la obra de Gill [5]).

disfrute por los autores de una beca IBM para la investigación en grupos. Después de la entrega de la Memoria final [6] al Comité Ejecutivo del Centro de Cálculo de la Universidad de Madrid hemos continuado el estudio. La simulación busca poner de relieve los aspectos matemáticos algorítmicos de la función de cada órgano. Para nosotros la cadena de comunicación se refleja en los pasos indicados en la figura 1.5, más algún programa de cálculo en el correspondiente cuerpo de Galois.

Para seguir esta comunicación suponemos en los presentes o lectores unos conocimientos de base en teoría matemática de códigos, circuitos secuenciales lineales y programación.

2. CODIFICACION

Las notaciones son:

$a(x) \equiv (a)$	polinomio de grado máximo $(k-1)$ o vector mensaje original, con k dígitos binarios.
$b(x) \equiv (b)$	polinomio de grado máximo $(n-1)$ o vector mensaje codificado, con n dígitos binarios.
$c(x) \equiv (c)$	polinomio de grado máximo $(n-k-1)$; resto de la división: $X^{n-k} \cdot a(x)/g(x)$.
$g(x)$	polinomio generador de grado $n-k = r$, siendo r el número de bits de control (igual a mt en el caso de códigos BCH).

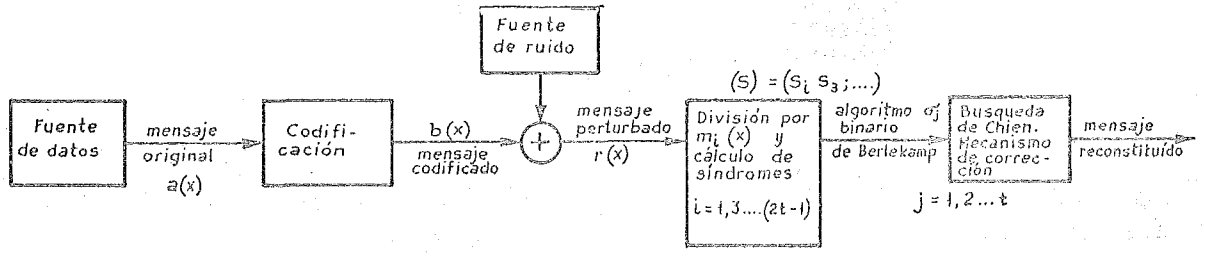

FIG. 1.5.

Cadena de funciones para la simulación.

La presente comunicación describe la simulación del funcionamiento de todos los órganos en una comunicación con códigos BCH. Este es un trabajo que se comenzó casi al final del

Las expresiones que siguen y el esquema funcional de la figura 2.1 ilustran el método de codificación que utilizamos (el cual genera una estructura separable, es decir: «vector mensaje

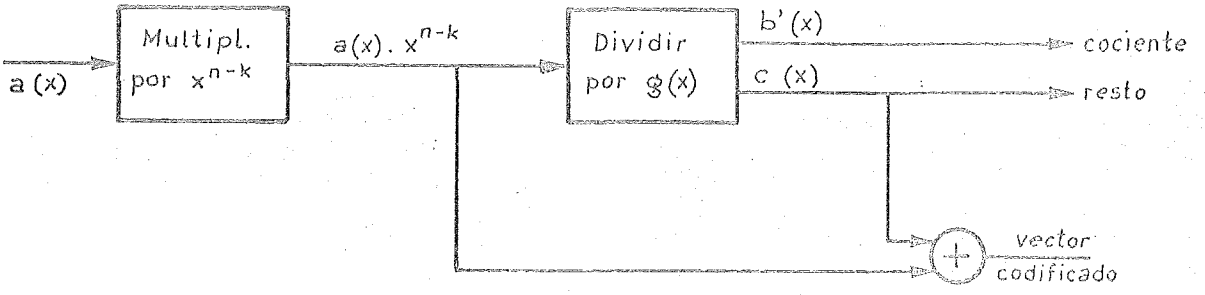

FIG. 2.1.

Esquema funcional para códigos separables.

codificado» = «vector mensaje original» + «bits de control»), ya que (—) y (+) equivalen en binario:

$$X^{n-k}\, a(x) = b'(x)\, g(x) + c(x)$$

$$b(x) = b'(x) \cdot g(x) = \underbrace{X^{n-k}\, a(x)}_{\text{vector original}} - \underbrace{c(x)}_{\text{bits de control}}$$

$b(x)$ es un polinomio múltiplo de $g(x)$, luego sus coeficientes pertenecen al código.

Desde el punto de vista de «circuitería», el problema se resuelve con el esquema de la figura 2.2 (cf. nuestra memoria [6]), donde las posiciones del conmutador corresponden a 0 para $t < k$ y a 1 para $t \geqslant k$, con $g_i \in \{0,1\}$.

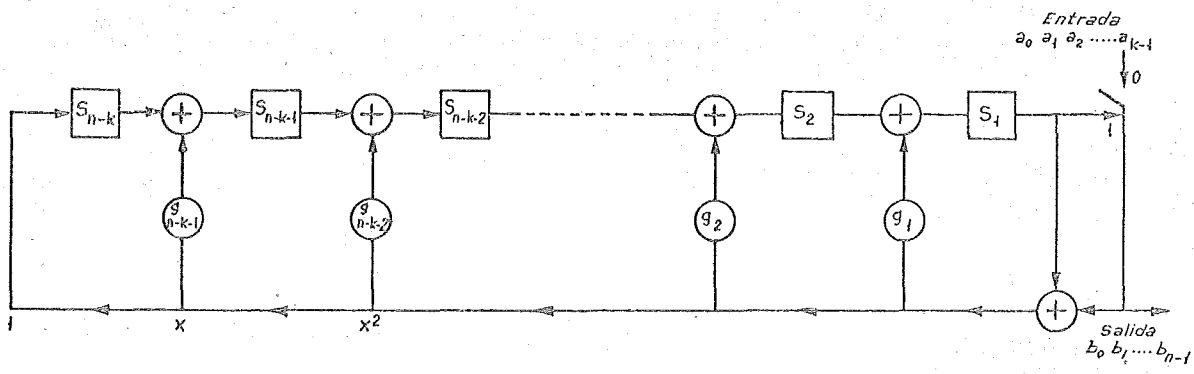

FIG. 2.2.

Circuito codificador de $(a_0\, a_1 \ldots a_{k-1})$ *en* $(b_0\, b_1 \ldots b_{n-1})$ *con estructura separable.*

El vector codificado es:

$$(b_0\, b_1 \ldots b_{n-1}) = (a_0\, a_1 \ldots a_{k-1}\, S_1 \ldots S_{n-k})$$

donde $(S_1\, S_1 \ldots S_{n-k})$, estado del circuito en el instante inmediato después de haber procesado los bits de información, constituyen los bits de control.

Desde el punto de vista de la simulación, necesitamos poner de manifiesto la estructura recurrente, implícita en el modelo polinómico. Llamamos s_j al estado del registro número j en un instante, t y s_j' al estado del mismo en el instante $t + 1$. En un instante cualquiera $t = i$, las ecuaciones recurrentes del sistema son:

$$\begin{cases} S'_1 = S_2 + (S_1 + a_{i-1})g_1 \\ S'_2 = S_3 + (S_1 + a_{i-1})g_2 \\ S'_3 = S_4 + (S_1 + a_{i-1})g_3 \\ \ldots\ldots\ldots\ldots\ldots \\ S'_{n-k-1} = S_{n-k} + (S_1 + a_{i-1})g_{n-k-1} \\ S'_{n-k} = (S_1 + a_{i-1}) \cdot 1 \end{cases}$$

3. GENERACION DE VECTORES RUIDO

La simulación consiste en reproducir el funcionamiento de la cadena, calculando con ordenador las transformaciones que sufre un mensaje cualquiera a causa de todos los posibles vectores ruido de 1, 2, 3, ... t bits de error. Una forma óptima de producir todos los vectores posibles de una determinada longitud se basa en una propiedad de los polinomios primitivos de grado n: «un elemento primitivo genera por potenciación sucesiva todos los elementos del grupo multiplicativo $GF(2^n)$» (fig. 3.1).

El comportamiento idealizado en la figura 3.1, donde las bolas representan elementos, es equivalente a la transformación lineal sobre un vector de longitud n, de componentes tomados en $\{0,1\}$, representada por una matriz

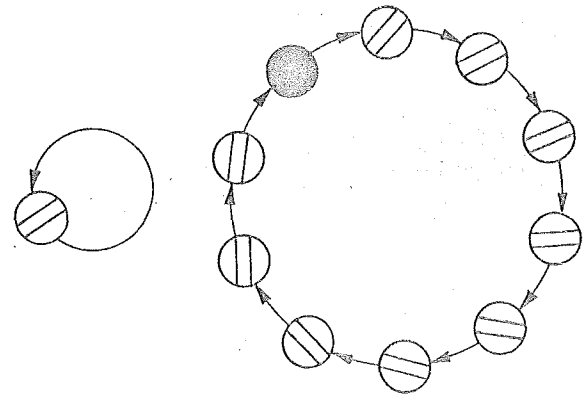

FIG. 3.1.

Recorrido del grupo multiplicativo $GF(2^n)$ *por potenciación de un elemento primitivo. A la izquierda, el elemento* 0.

$$A = \begin{pmatrix} 0 & 1 & & & & \\ & 0 & 1 & & & \\ & & \ddots & \ddots & & \\ & & & \ddots & \ddots & \\ & & & & \ddots & 1 \\ -\alpha_0 & -\alpha_1 & -\alpha_2 & \cdots & & -\alpha_{n-1} \end{pmatrix}$$

y realizada por el circuito de la figura 3.2. La evolución autónoma de este circuito es similar a la reflejada en la figura 3.1, si se identifica el

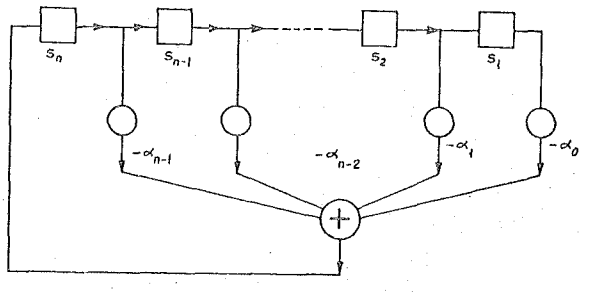

FIG. 3.2.

Circuito autónomo que genera los elementos especificados por el polinomio $\alpha_0 + \alpha_1 x + \ldots + \alpha_{n-1} x^{n-1} + x^n$.

vector estado (S_1; S_2; S_3; ... S_n) con cada una de las bolas y se toma como estado de partida (bola más oscura) un vector cualquiera diferente de (0, 0, 0, ... 0).

Se supone $\alpha_0 + \alpha_1 x \ldots + \alpha_{n-1} x^{n-1} + x^n$ polinomio primitivo.

La ecuación recurrente, calculada al nivel de cada registro, es:

$$\begin{cases} S'_n = -\alpha_{n-1} S_n - \alpha_{n-2} S_{n-1} - \ldots - \alpha_1 S_2 - \alpha_0 S_1 \\ S'_{n-1} = S_n \\ S'_{n-2} = S_{n-1} \\ \ldots\ldots\ldots \\ S'_1 = S_2 \end{cases}$$

Tomamos, como elemento de partida, el estado (1, 0, 0, ... 0), ya que es indiferente. Con objeto de generar sólo los vectores ruido que nos pueden interesar, con i bits de error por ejemplo, sometemos a cada uno de los que produce el programa a un test consistente en la suma de sus bits de valor 1, rechazándolo cuando la suma es diferente de i.

4. DECODIFICACION

Utilizamos como esquema decodificador la división por un polinomio generador $g(x)$ que, en el caso presente, se descompone en divisiones en paralelo por los polinomios mínimos $m_i(x)$, factores de $g(x)$. El circuito de división, cuyo resto es el estado de los registros después de procesar los n bits del mensaje perturbado, se especifica en la figura 4.1.

Las ecuaciones recurrentes son:

$$\begin{cases} S'_1 = r_{i-1} + S_1 \\ S'_{i-1} = S_i + g_{i-1} \cdot S_1 \\ S'_{i-2} = S_{i-1} + g_{i-2} \cdot S_1 \\ \ldots\ldots\ldots\ldots \\ S'_2 = S_3 + g_2 \cdot S_1 \\ S'_1 = S_2 + g_1 \cdot S_1 \end{cases}$$

Conocido el resto de la división por el polinomio mínimo $m_i(x)$ es preciso valorarlo para las potencias α^j del elemento primitivo del cuerpo $GF(2^m)$, ya que los síndromes son:

$$S_j = r(\alpha^j)$$

La valoración se efectúa dentro del mismo programa de decodificación en su primera fase.

5. ALGORITMO ABREVIADO PARA OBTENER EL POLINOMIO DE POSICION DE ERROR

Encontrados los síndromes, el mayor problema es encontrar las posiciones de error x_1, x_2, ... x_e de las ecuaciones

$$\sum_{i=}^{e} x_i^j = S_j \quad , \quad j = 1, 2, 3, \ldots, 2t$$

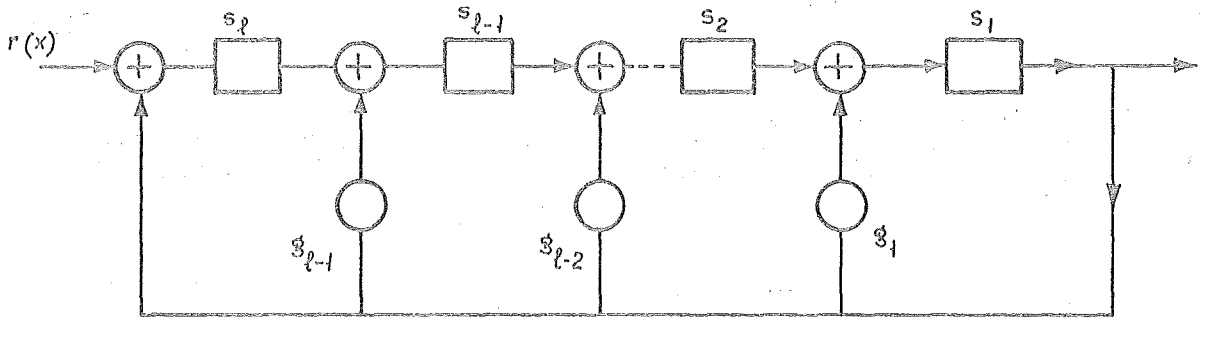

FIG. 4.1.

Circuito divisor del polinomio r(x) *por el polinomio* g(x).

Estas ecuaciones pueden tener varias soluciones, cada una correspondiente a los diferentes tipos de error en la misma clase del grupo aditivo de las palabras código.

Con el fin de resolver estas ecuaciones, se define el polinomio localizador de error de la siguiente forma:

$$\sigma(z) = \prod_{i=1}^{e} (1 - x_i z) = 1 + \sum_{j=1}^{e} \sigma\ z^j$$

Para relacionar las σ con las S, introducimos la función generatriz

$$S(z) = \sum_{j=1}^{\infty} S_j z^j = \sum_{j=1}^{\infty} \sum_{i=1}^{e} x_i^j z^j = \sum_{i=1}^{e} \frac{x_i z}{1 - x_i z}$$

la cual se convierte en

$$[1 + S(z)]\, \sigma(z) = \sigma(z) + \sum_{i=1}^{e} x_i z \prod_{j \neq i} (1 - x_j z)$$

Como el decodificador sólo conoce los coeficientes de las potencias en z de $S(z)$ de grado menor o igual a $2t$, obtenemos la ecuación fundamental

$$[1 + S(z)]\, \sigma(z) \equiv \omega(z) \bmod (z^{2t+1})$$

donde

$$\omega(z) = \sigma(z) + \sum_{i=1}^{e} x_i z \prod_{j \neq 1} (1 - x_j z)$$

Considerando las ecuaciones

$$(1 + S)\, \sigma^{(k)} = \omega^{(k)} \bmod z^{k+1}$$

para $k = 0, 1, 2, \ldots, 2t$, encontraremos polinomios

$$\sigma^{(k)} = \sum_i \sigma^{(k)} z^i \quad \text{y} \quad \omega^{(k)} = \sum_i \omega_i^{(k)} z^i$$

que resuelven esta ecuación.

Para códigos BCH, Berlekamp obtuvo un algoritmo reducido que permite obtener las $\sigma^{(k)}$. El algoritmo es el siguiente:

1.º $\sigma^{(0)} = 1$
$\tau^{(0)} = 1$

2.º Si S_{2k+1} es desconocido, «stop»

3.º $\Delta_1^{(2k)}$ igual al coeficiente de z^{2k+} en el producto $(1 + S) \cdot \sigma^{(2k)}$

4.º $\sigma^{(2k+2)} = \sigma^{(2k)} + \Delta_1^{(2k)}\, z\, \tau^{(2k)}$

5.º $\tau^{(2k+2)} = \begin{cases} z^2\, \tau^{(2k)} \text{ si } \Delta_1^{(2k)} = 0 \text{ ó si } \delta(\sigma^{(2k)}) > k\ (1) \\ \dfrac{z\, \sigma^{(2k)}}{\Delta_1^{(2k)}} \text{ si } \Delta_1^{(2k)} \neq 0 \quad \text{y} \quad \delta(\sigma^{(2k)}) \leqslant k\ (2) \end{cases}$

donde $\tau^{(2k)}$ son polinomios auxiliares.

Para facilitar la terminología utilizada a lo largo de esta comunicación llamaremos ordinal de un elemento del cuerpo de Galois a la potencia aumentada en una unidad de la potencia del elemento generador del cuerpo que sea igual al citado elemento. Haremos el convenio de que el ordinal de 1 sea 1 y el de 0 sea igual al número de elementos del cuerpo; en el programa se llamará T ∅ PE1.

Para poder realizar sumas y productos de elementos hemos escrito unas funciones llamadas SUMA y PRODUC, respectivamente.

La función SUMA necesita como variables de entrada el ordinal de los dos elementos que se desean sumar, el número de componentes y el ordinal del 0. La salida de esta función es el ordinal del resultado. Esta función comparte con el programa principal una zona de memoria, que está compuesta por una matriz formada por todos los elementos del cuerpo, y otra que da, por tabla de doble entrada, la suma módulo 2.

La función PRODUC tiene como entradas las ordinales de los dos elementos que se quiere multiplicar y el ordinal del 0. Su salida es el ordinal del resultado.

Para este algoritmo se suponen conocidas las sumas S_1, S_2, S_4, ..., de las potencias de las posiciones de error y la tabla de elementos del cuerpo dada por su ordinal y con todas sus componentes; también necesita las funciones SUMA, PRODUC y la suma módulo 2. Como resultado se obtiene $\sigma^{(2k)}$, como una matriz rectangular en la que el primer índice es igual a $k + 1$, y el segundo menos una unidad da el exponente de z en la función $\sigma^{(2k)}$; esta matriz recibe el nombre de SIGMA.

Si no se da el término S_{2k+1} el programa se para, indicando con ello que el cálculo ha terminado.

6. METODO DE CORRECCION CICLICA (CHIEN)

El fundamento del método de Chien es el siguiente:

Dado el polinomio localizador $\sigma(z)$ se resolverá secuencialmente la ecuación $\sigma(z^{-1}) = 0$, comprobando para cada dígito o bit de salida si el elemento x_i del cuerpo que corresponde a su posición es o no raíz, esto es, si $\sigma(x_i^{-1})$ es o no igual a cero. Si $\sigma\ (x_i^{-1}) \neq 0$ este dígito sale de la memoria tampón sin variación. En caso contrario, $\sigma(x_i^{-1}) = 0$, se corrige el bit al salir de la memoria tampón. De este modo se van probando los dígitos de salida, cuyas posiciones son, respectivamente, α^{-1}, α^{-2}, ... 1.

Conocida la ecuación $\sigma(x) = 1 + \sigma_1 x + \ldots + \sigma_t x^t$, se calcula $\sigma(\alpha)$, $\sigma(\alpha^2)$, ... $\sigma(\alpha^k)$ ... en desplazamientos sucesivos. En el intervalo k se obtiene el valor $\sigma(\alpha^k)$. Si es nulo, α^{-k} es una raíz recíproca al polinomio localizador por el que se corrige el bit que está en la posición α^{-k} y que en este momento sale de la memoria tampón. Este es el procedimiento que se ha seguido en la simulación. Hemos preparado un programa para la evaluación de los valores de $\sigma(\alpha^k)$ ($k = 1, \ldots t$). Dado que los coeficien-

tes σ_k pertenecen al mismo cuerpo que las potencias de α, el programa trabaja con los exponentes de los términos $\sigma_i\ \alpha^{ik}$ y emplea las subrutinas para la suma de elementos que ya se han mencionado.

AGRADECIMIENTO

Los autores desean expresar su agradecimiento al Comité Ejecutivo del Centro de Cálculo de la Universidad de Madrid por la concesión de una Beca de Investigación del fondo IBM, que les ha permitido profundizar los estudios sobre la teoría Matemática de los Códigos y Sistemas de Codificación.

A la señorita D. Pachón, por su entusiasta y vital colaboración en la preparación y puesta a punto de los programas para el ordenador IBM 70 90 del CCUM.

REFERENCIAS

[1] W. W. Peterson: «Error-correcting Codes». J. Wiley, New York, 1961.

[2] R. T. Chien: «Cyclic decoding procedures for Bose-Chaudhuri-Hocquenghem Codes». IEEE, IT-10 núm. 4, 1964.

[3] E. R. Berlekamp: «On decoding binary BCH Codes». IEEE, IT-11, núm. 4, 1965.

[4] E. R. Berlekamp: «Algebraic Coding theory». McGraw Hill, 1968.

[5] A. Gill: «Linear sequential circuits». McGraw Hill, 1966.

[6] F. Sáez Vacas, J. M. Hernando Rábanos, B. Fontana Sanchis: «Teoría Matemática de los Códigos. Aportaciones a la investigación de los mecanismos de corrección de error, en particular de los códigos BCH». Memoria final para el Centro de Cálculo de la Universidad de Madrid. Enero, 1970.