CODIGOS DE DETECCION DE ERRORES POR CHEQUEO REDUNDANTE CICLICO (CRC)

Medina Alicia
Montalvo Luis, Ing.
ESCUELA POLITECNICA NACIONAL

RESUMEN

Se presenta un estudio comparativo de los distintos algoritmos de cálculo del código para Chequeo Redundante Cíclico (CRC) utilizado para la detección de errores en sistemas de transmisión de datos. Se analizan brevemente los principios básicos de cada uno de los algoritmos y se compara el tiempo de ejecución y el área de memoria requerida por los programas respectivos.

ABSTRACT

A comparative study of three different algorithms for computing the Cyclic Redundancy Check (CRC) used in error detection in data transmission systems is presented. The basic principles of each one of these algorithms are briefly analized and the execution time and memory area required by the corresponding programs are compared.

INTRODUCCION

Un sistema de transmisión y almacenamiento de información debe tener como características alta confiabilidad y eficiencia, por lo que en su diseño una de las preocupaciones es disponer del control de errores adecuado que permita recibir los datos de la fuente sin ninguna alteración.

Existen diferentes códigos que permiten la detección de errores, uno de ellos es el de detección de errores por chequeo redundante cíclico (CRC), que consiste en añadir al mensaje a transmitir dos bytes de redundancia. Se ha estudiado y aplicado de manera experimental, en un proceso de transmisión de datos entre dos computadores, tres algoritmos distintos para el cálculo del CRC, los que se diferencian en su velocidad de procesamiento del mensaje y en el área de memoria que ocupan los programas correspondientes.

Los códigos de chequeo redundante cíclico son una composición de chequeo de paridad con un código cíclico. Para éstos se tienen algunos polinomios generadores, en este trabajo se utiliza el de la Recomendación CCITT V.41: $X^{16} + X^{12} + X^5 + 1$.

Con el CRC es posible detectar errores aleatorios o del tipo ráfaga, es decir en secuencias consecutivas de bits.

1 Códigos Cíclicos

Los códigos cíclicos constituyen una subclase de los códigos bloque en los cuales un mensaje representado por la k_tupla:

$$u = (u_1, u_2, \ldots u_k)$$

llamada vector de código se transforma en el codificador en una n_tupla:

$$v=(v_1, v_2, \ldots \ldots v_n)$$

que se denomina palabra de código. Donde $1 \leq i \leq n$ es igual a 1 o 0.

Por tanto, a los 2^k posibles mensajes le corresponden 2^k palabras de código, que forman el código bloque C(n,k).

Los códigos bloque pueden ser lineales, es decir que la suma de dos palabras de código es otra palabra de código.

1.1 Definición y Representación de los Códigos Cíclicos

Un código lineal C(n,k) se denomina cíclico, si con cada permutación circular de un vector de código en C se obtiene otro vector también en C. Así, para el vector:

$$v= (v_0, v_1, \ldots v_{n-1})$$

con un desplazamiento hacia la derecha se obtiene:

$$v^{(1)}= (v_{n-1}, v_0, \ldots v_{n-2})$$

Los componentes de v pueden tratarse como coeficientes de un polinomio de grado menor o igual a n-1:

$$v(X)= v_0 + v_1X + v_2X^2 + \ldots + v_{n-1}X^{n-1} \quad (1.1)$$

con v_i= 0 o 1. v(X) se conoce como polinomio de código.

Después de "i" desplazamientos cíclicos se tiene:

$$v^{(i)}(X)= v_{n-i} + v_{n-i+1}X + \ldots + v_{n-1}X^{i-1} + v_0X^i + v_1X^{i+1} + \ldots v_{n-i-1} X^{n-1} \quad (1.2)$$

1.2 Propiedades de los Códigos Cíclicos

Considerando un polinomio de grado mínimo r

$$g(X)= g_0 + g_1X + \ldots + g_{r-1}X^{r-1} + X^r$$

puede demostrarse que tiene el coeficiente g_0 igual a 1, quedando:

$$g(X)= 1 + g_1X + \ldots + g_{r-1}X^{r-1} + X^r \quad (1.3)$$

Un polinomio binario v(X) de grado menor o igual a "n-1" es un polinomio de código si es múltiplo de g(X), lo que se expresa de la siguiente manera:

$$v(X)= a(X)\ g(X)$$

Donde a(X) es de grado menor o igual a "n-r-1".

Los coeficientes de a(X) pueden ser 0 o 1, el número de polinomios de grado menor o igual a "n-1" múltiplos de g(X) es 2^{n-r}. Para una longitud k se pueden tener 2^k posibles mensajes diferentes, a los que les

corresponderán los polinomios de código anteriores, por lo tanto, $2^{n-r} = 2^k$, y $r=n-k$. Así:

$$v(X)= u(X)g(X) \qquad (1.4)$$

$$v(X)= (u_0 +u_1X +...+u_{k-1}X^{k-1})\ g(X)$$

Si los coeficientes de $u(X)$, u_0 ,u_1 ,...u_{k-1} son los k dígitos de la información; $v(X)$ es el correspondiente polinomio de código. La codificación puede realizarse multiplicando el mensaje $u(X)$ por $g(X)$.

Por lo tanto, un código cíclico es especificado completamente por el polinomio de grado mínimo $g(X)$ dado en (1.4) llamado **polinomio generador** del código, cuyo grado es igual al número de dígitos de chequeo de paridad.

1.3 Codificación de Códigos Cíclicos

Para la codificación puede utilizarse la relación $v(X)= u(X)\ g(X)$. Realizando la multiplicación (en módulo-2) del mensaje por el polinomio generador se obtiene la palabra codificada, pero en este caso no siempre se conservan los dígitos del mensaje sin alteración, lo que es una característica favorable de los códigos sistemáticos.

Código Cíclico Sistemático

Dado el polinomio generador $g(X)$ de un código cíclico (n,k), el código obtenido puede tener forma sistemática, **es decir que k dígitos del vector de código corresponden a la información inalterada, y n-k bits son de chequeo de paridad.**

Si el mensaje a ser codificado es $u= (u_0 ,u_1 ,...,u_{k-1})$, le corresponde el polinomio de mensaje:

$$u(X)= u_0 +u_1X +...+u_{k-1}X^{k-1}$$

Multiplicando $u(X)$ por X^{n-k} y luego dividiendo por $g(X)$ se obtiene un polinomio de grado menor o igual a n-1:

$$X^{n-k}\ u(X)= a(X)g(X)+b(X) \qquad (1.5)$$

donde $a(X)$ y $b(X)$ son el cociente y residuo respectivamente.

Siendo el grado del polinomio generador n-k, $b(X)$ debe ser de grado menor o igual a n-k-1 Modificando (1.5):

$$b(X)+X^{n-k}\ u(X)= a(X)g(X) \qquad (1.6)$$

Este polinomio es múltiplo de $g(X)$ y por tanto pertenece al código cíclico generado por $g(X)$. Así se tiene:

$$b(X)+X^{n-k}\ u(X)= b_0 +b_1X +.....+b_{n-k-1}X^{n-k-1} + u_0X^{n-k} +u_1X^{n-k+1}+.....+u_{k-1}X^{n-1} \qquad (1.7)$$

que corresponde al vector de código:

$$(b_0 ,b_1 ,...,b_{n-k-1} ,u_0 ,u_1 ,....u_{k-1}).$$

Donde hay k dígitos de información y n-k dígitos de chequeo de paridad que son el residuo de dividir $X^{n-k}\ u(X)$ por $g(X)$.

El circuito de codificación para el código cíclico basándose en el polinomio generador es el que se muestra en la Fig. 1.1.

1.4 Cálculo del Síndrome y Detección del Error

Si se transmite un vector v y se recibe $r=(r_0,r_1,.r_{n-1})$; debido a que el canal de comunicaciones tiene ruido estos dos vectores pueden no ser iguales.

El primer paso para decodificar un código lineal es calcular el síndrome, que es $s=r.H^T$, donde "H" es la matriz de chequeo de paridad. Si el síndrome es cero quiere decir que "r" es un vector de código y el decodificador lo acepta como el transmitido, pero si el síndrome es diferente de cero indica que "r" no es un vector de código y un error ha sido detectado.

Para un código lineal sistemático, el síndrome es simplemente el vector suma de los dígitos de paridad recibidos y los recalculados a partir de los bits de información que llegan al receptor.

Si el vector recibido se trata como un polinomio de grado menor o igual a n-1 y se divide por $g(X)$ se tiene:

$$r(X)= r_0 +r_1X +r_2X^2 +....+r_{n-1}X^{n-1}$$

$$r(X)= a(X)g(X)+s(X) \qquad (1.8)$$

donde el residuo $s(X)$ es un polinomio de grado igual o menor a n-k-1. Los n-k coeficientes de $s(X)$ forman el síndrome.

Un circuito de división igual al empleado para codificación con n-k etapas de desplazamiento puede emplearse para calcular el síndrome, con la diferencia que el polinomio recibido $r(X)$ es desplazado en el circuito desde el lado izquierdo.

2 Algoritmos de Cálculo del CRC

Si se transmiten x bytes por mensaje, añadiendo el control con CRC se tienen en total x+2 bytes.

Siguiendo el procedimiento de obtención del código cíclico sistemático (Ec. 1.7), para el polinomio de mensaje $u(X)$, y siendo $r=n-k$ se obtiene:

$$\frac{X^r\ u(X)}{g(X)} = a(X) + \frac{b(X)}{g(X)}$$

donde $a(X)$ es el cociente, $b(X)$ el residuo y + es el símbolo de suma módulo 2 (XOR).

$$X^r\ u(X)= a(X)\ g(X) + b(X)$$

$$X^r\ u(X) + b(X)= a(X)\ g(X) \qquad (1.9)$$

El algoritmo del CRC calcula $b(X)$. En (Ec. 1.9) se observa que los bits transmitidos (información y control) siguen siendo divisibles por $g(X)$ si y sólo si los bits no han sido alterados. De esta manera, si en el receptor vuelven a dividirse el mensaje y CRC recibidos por $g(X)$, cuando el resultado es cero se asume que no hay errores o son no detectables, si es uno se considera que hay error.

Para el calculo en hardware del CRC-CCITT con polinomio generador $1+X^5+X^{12}+X^{16}$ se tiene el circuito de la Fig. 1.2.

Se puede realizar la emulación en software del funcionamiento en hardware, utilizando el procedimiento detallado en la Tabla 2.1:

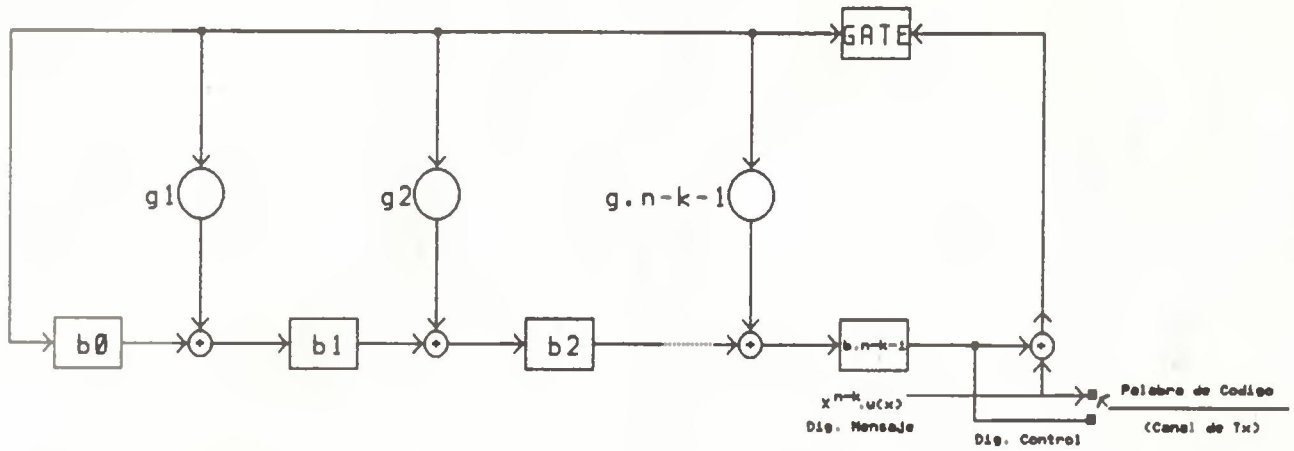


Fig.1.1: CODIFICADOR DE CODIGO CICLICO c(n,k)

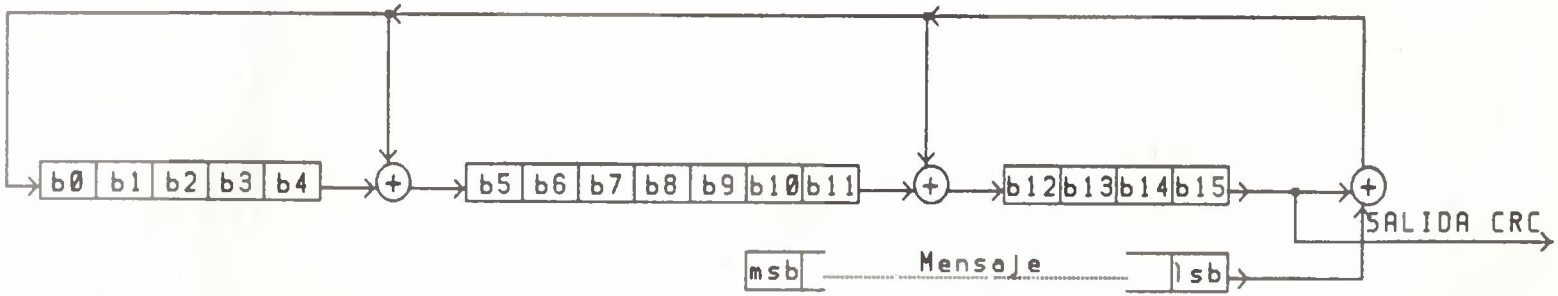


Fig. 1.2: CIRCUITO DE CALCULO DEL CRC-CCITT

Tabla 2.1 Procedimiento de Cálculo del CRC del Mensaje

SH	IN	X^0	X^1	X^2	X^3	X^4	X^5	X^6	X^7	X^8	X^9	X^{10}	X^{11}	X^{12}	X^{13}	X^{14}	X^{15}
		b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}	b_{11}	b_{12}	b_{13}	b_{14}	b_{15}
0		c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}
1	u_7	c_{15} u_4	c_0	c_1	c_2	c_3	c_4 c_{15} u_7	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11} c_{15} u_3	c_{12}	c_{13}	c_{14}
2	u_6	c_{14} u_6	c_{15} u_3	c_0	c_1	c_2	c_3 c_{14} u_6	c_4 c_{15} u_3	c_5	c_6	c_7	c_8	c_9	c_{10} c_{14} u_6	c_{11} c_{15} u_3	c_{12}	c_{13}
3	u_5	c_{13} u_5	c_{14} u_6	c_{15} u_3	c_0	c_1	c_2 c_{13} u_5	c_3 c_{14} u_6	c_4 c_{15} u_3	c_5	c_6	c_7	c_8	c_9 c_{13} u_5	c_{10} c_{14} u_6	c_{11} c_{15} u_3	c_{12}
⋮		⋮					⋮							⋮			⋮
8	u_0	c_8 c_{12} u_4 u_0	c_9 c_{13} u_5 u_1	c_{10} c_{14} u_6 u_2	c_{11} c_{15} u_7 u_3	c_{12} u_4	c_{13} u_5 c_8 c_{12} u_4 u_0	c_{14} u_6 c_9 c_{13} u_5 u_1	c_{15} u_7 c_{10} c_{14} u_6 u_2	c_0 c_{11} c_{15} u_7 u_3	c_1 c_{12} u_4	c_2 c_{13} u_5	c_3 c_{14} u_6	c_4 c_{15} u_7 c_8 c_{12} u_4 u_0	c_5 c_9 c_{13} u_5 u_1	c_6 c_{10} c_{14} u_6 u_2	c_7 c_{11} c_{15} u_7 u_3

La notación utilizada en la Tabla 2.1 es la siguiente:

a) Los bits del mensaje se numeran de 0 a 7 y los del CRC de 0 a 15, siendo 7 y 15 los menos significativos.

b) Los desplazamientos se realizan hacia la derecha.

c) SH es la columna con el número de desplazamiento.

d) IN es la columna con el bit de entrada u_i, con $0 \leq i \leq 7$.

e) b_i es el i-ésimo bit del CRC, donde $0 \leq i \leq 15$.

f) c_i es el i-ésimo bit del CRC inicial, con $0 \leq i \leq 15$.

g) Los elementos de las columnas se suman en módulo-2.

h) Las realimentaciones en el circuito de hardware corresponden a las potencias del polinomio, es decir, X^0, X^5, X^{12}.

2.1 Algoritmo de cálculo del CRC bit a bit

De la Tabla 2.1 se obtiene el algoritmo:

1. Realizar la suma or-exclusivo del bit menos significativo del CRC con el bit de mensaje.

2. Colocar el resultado en las posiciones de las potencias de X^0, X^5, X^{12}.

3. Desplazar la palabra del CRC un bit a la derecha.

4. Repetir los pasos 1 a 3 para cada bit.

El CRC resultante es la suma en módulo-2 de los términos obtenidos para el último bit de mensaje.

2.2 Algoritmo de Cálculo del CRC por Byte con Búsqueda en una Tabla

Puesto que los microcomputadores trabajan con bytes, se trata de obtener un algoritmo que calcule el CRC por bytes, con el mismo resultado que después de ocho desplazamientos bit a bit. Para ésto, se define un vector X como sigue:

$$X = (X_0, X_1, X_2, X_3, X_4, X_5, X_6, X_7)$$

$$\begin{aligned}
X_0 &= c_8 + u_0 \\
X_1 &= c_9 + u_1 \\
X_2 &= c_{10} + u_2 \\
X_3 &= c_{11} + u_3 \\
X_4 &= c_{12} + u_4 \\
X_5 &= c_{13} + u_5 \\
X_6 &= c_{14} + u_6 \\
X_7 &= c_{15} + u_7
\end{aligned}$$

y aplicando la propiedad conmutativa A + B = B + A se llega a la Tabla 2.2. La cual al ser reordenada genera el resultado mostrado en la Tabla 2.3.

En la Tabla 2.3 se observa que el byte más significativo del CRC se desplaza a la parte menos significativa y se suma con una palabra de 16 bits (combinación de los Xi), que es el CRC de un byte. Hay 256 posibles valores para el vector X y con los CRC correspondientes se encuentran recopilados en la Tabla 2.4.

Utilizando la Tabla 2.4 de los CRC de mensajes de un byte se llega al siguiente algoritmo.

1. Realizar la suma or-exclusivo del byte menos significativo del CRC y el byte de mensaje, éste es el vector X.

Tabla 2.2: CRC de Palabras de 8 Bits

SH	IN																
		b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}	b_{11}	b_{12}	b_{13}	b_{14}	b_{15}
8	u_0	X_0 X_4	X_1 X_5	X_2 X_6	X_3 X_7	X_4	X_5 X_0 X_4	X_6 X_1 X_5	X_7 X_2 X_6	c_0 X_3 X_7	c_1 X_4	c_2 X_5	c_3 X_6	c_4 X_7 X_0 X_4	c_5 X_1 X_5	c_6 X_2 X_6	c_7 X_3 X_7

Tabla 2.3 Contenido Final del CRC para Palabras de 8 Bits

SH	IN																
		b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}	b_{11}	b_{12}	b_{13}	b_{14}	b_{15}
		0	0	0	0	0	0	0	0	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7
		X_0 X_4	X_1 X_5	X_2 X_6	X_3 X_7	X_4	X_5 X_0 X_4	X_6 X_1 X_5	X_7 X_2 X_6	X_3 X_7	X_4	X_5	X_6	X_7 X_4 X_0	X_5 X_1	X_6 X_2	X_7 X_3

Tabla 2.4: CRC Para Mensajes de un Byte (Polinomio Generador: $X^{16} + X^{12} + X^{5} + 1$)

	0	1	2	3	4	5	6	7
1	0000h	1189h	2312h	329Bh	4624h	57ADh	6536h	74BFh
2	8C48h	9DC1h	AF5Ah	BED3h	CA6Ch	DBE5h	E97Eh	F8F7h
3	1081h	0108h	3393h	221Ah	56A5h	472Ch	75B7h	643Eh
4	9CC9h	8D40h	BFDBh	AE52h	DAEDh	CB64h	F9FFh	E876h
5	2102h	308Bh	0210h	1399h	6726h	76AFh	4434h	55BDh
6	AD4Ah	BCC3h	8E58h	9FD1h	EB6Eh	FAE7h	C87Ch	D9F5h
7	3183h	200Ah	1291h	0318h	77A7h	662Eh	54B5h	453Ch
8	BDCBh	AC42h	9ED9h	8F50h	FBEFh	EA66h	D8FDh	C974h
9	4204h	538Dh	6116h	709Fh	0420h	15A9h	2732h	36BBh
10	CE4Ch	DFC5h	ED5Eh	FCD7h	8868h	99E1h	AB7Ah	BAF3h
11	5285h	430Ch	7197h	601Eh	14A1h	0528h	37B3h	263Ah
12	DECDh	CF44h	FDDFh	EC56h	9BE9h	8960h	BBFBh	AA72h
13	6306h	728Fh	4014h	519Dh	2522h	34ABh	0630h	17B9h
14	EF4Eh	FEC7h	CC5Ch	DDD5h	A96Ah	B8E3h	8A78h	9BF1h
15	7387h	620Eh	5095h	411Ch	35A3h	242Ah	16B1h	0738h
16	FFCFh	EE46h	DCDDh	CD54h	B9EBh	A862h	9AF9h	8B70h
17	8408h	9581h	A71Ah	B693h	C22Ch	D3A5h	E13Eh	F0B7h
18	0840h	19C9h	2B52h	3ADBh	4E64h	5FEDh	6D76h	7CFFh
19	9489h	8500h	B79Bh	A612h	D2ADh	C324h	F1BFh	E036h
20	18C1h	0948h	3BD3h	2A5Ah	5EE5h	4F6Ch	7DF7h	6C7Eh
21	A50Ah	B483h	8618h	9791h	E32Eh	F2A7h	C03Ch	D1B5h
22	2942h	38CBh	0A50h	18D9h	6F66h	7EEFh	4C74h	5DFDh
23	B58Bh	A402h	9699h	8710h	F3AFh	E226h	D0BDh	C134h
24	39C3h	284Ah	1AD1h	0B58h	7FE7h	6E6Eh	5CF5h	4D7Ch
25	C60Ch	D785h	E51Eh	F497h	8028h	91A1h	A33Ah	B2B3h
26	4A44h	5BCDh	6956h	78DFh	0C60h	1DE9h	2F72h	3EFBh
27	D68Dh	C704h	F59Fh	E416h	90A9h	8120h	B3BBh	A232h
28	5AC5h	4B4Ch	79D7h	685Eh	1CE1h	0D68h	3FF3h	2E7Ah
29	E70Eh	F687h	C41Ch	D595h	A12Ah	B0A3h	823Bh	93B1h
30	6B46h	7ACFh	4854h	59DDh	2D62h	3CEBh	0E70h	1FF9h
31	F78Fh	E606h	D49Dh	C514h	B1ABh	A022h	92B9h	8330h
32	7BC7h	6A4Eh	58D5h	495Ch	3DE3h	2C6Ah	1EF1h	0F78h

2. Utilizar el vector X como índice de la tabla y buscar el CRC que le corresponde.

3. Desplazar el CRC ocho bits a la derecha.

4. Hacer la suma or-exclusivo de los resultados de 2 y 3.

5. Repetir el proceso para cada byte de mensaje.

2.3 Algoritmo de cálculo del CRC por byte "on the fly"

El CRC de un byte puede calcularse también dentro del procedimiento, evitando así almacenar la tabla como se indica en el siguiente algoritmo.

1. Realizar la suma or-exclusivo del byte menos significativo del CRC y el byte de mensaje (vector X).

2. Calcular la combinación de los X_i de la tabla 1.3 para los componentes de X.

3. Desplazar el CRC ocho bits a la derecha.

4. Hacer la suma or-exclusivo de los resultados de 2 y 3.

5. Repetir el proceso para cada byte del mensaje.

El **CRC inicial** para los algoritmos es igual a cero.

2.3 Ejemplo de cálculo del CRC

Mensaje: 0,1,3
Mensaje en hexadecimal: 00h,01h,03h
a) Algoritmo de cálculo bit a bit.
Datos: 00000000,00000001,00000011

			Dato	u_i+C_{Lsb}
00000	**0000000**	**0000**		
00000	0000000	0000	0	0
00000	0000000	0000	0	0
00000	0000000	0000	0	0
00000	0000000	0000	0	0
00000	0000000	0000	0	0
00000	0000000	0000	0	0
00000	0000000	0000	0	0
00000	0000000	0000	0	0

CRC_1=0000h

10000	1000000	1000	1	1
01000	0100000	0100	0	0
00100	0010000	0010	0	0
00010	0001000	0001	0	0
10001	1000100	1000	0	1
01000	1100010	0100	0	0
00100	0110001	0010	0	0
00010	0011000	1001	0	0

CRC_2=1189h

00001	0001100	0100	1	0
10000	0000110	1010	1	1
01000	0000011	0101	0	0
10100	1000001	0010	0	1
01010	0100000	1001	0	0
10101	1010000	1100	0	1
01010	1101000	0110	0	0
00101	0110100	0011	0	0

CRC_3=2B43h

b) Algoritmo de cálculo por bytes.
Datos: 00000000,00000001,00000011

CRC Inicial= 0000h

```
0000  0000  0000  0000   0000h
XOR         0000  0000    00h  (Byte mens.)
-----------------------
0000  0000  0000  0000   00h=0d Indice

      Tab (0+1)= 0000h

0000  0000  0000  0000   0000h
XOR         0000  0000    00h
-----------------------
0000  0000  0000  0000   CRC1 = 0000h

0000  0000  0000  0000   0000h
XOR         0000  0001    01h  (Byte mens.)
-----------------------
0000  0000  0000  0001   01h=1d Indice

 Tab (1+1)= 1189h

0001  0001  1000  1001   11B9h
XOR         0000  0000    00h
-----------------------
0001  0001  1000  1001   CRC2 = 1189h

0001  0001  1000  1001   11B9h
XOR         0000  0011    03h  (Byte mens.)
-----------------------
0001  0001  1000  1010   8Ah=138d Indice

 Tab (138+1)= 2B52h

0010  1011  0101  0010   2B52h
XOR         0001  0001    11h
-----------------------
0010  1011  0100  0011   CRC3 = 2B43h
```

Los valores del CRC de un byte se consultaron en la Tabla 2.4.

3 Resultados Experimentales

Se ha desarrollado un programa didáctico que permite observar los resultados del cálculo del CRC y el tiempo de ejecución para los tres algoritmos. En la recepción se realiza la detección de errores, con retransmisión de los bloques de mensaje que resulten erróneos.

La simulación de los errores que pueden ocurrir en la línea se realiza modificando el mensaje original.

El mensaje recibido se graba en un archivo y se además se lee en la pantalla.

Se realizaron programas para obtener el tiempo de cálculo con los algoritmos para diferentes longitudes de mensaje, así como para conocer el área de memoria que requieren los programas respectivos. Los resultados se presentan en los gráficos 3.1 a 3.3, donde se tiene la siguiente notación:

Método 1: Cálculo bit a bit.
Método 2: Cálculo por byte con búsqueda en una tabla.
Método 3: Cálculo "on the fly".

De los gráficos se desprende lo siguiente:

1. La relación del tiempo con la longitud del mensaje es lineal para los tres algoritmos.

2. Con el método bit a bit se necesita un tiempo 4.86 veces mayor que con el método "on the fly" y 7.7 veces más que con el de búsqueda en una tabla.

3. El tiempo con el algoritmo "on the fly" es 1.6 veces mayor que con búsqueda en una tabla.

4. Almacenando la tabla de CRCs se necesita un 30% más de memoria que con el cálculo bit a bit y el 38% más que con el algoritmo "on the fly".

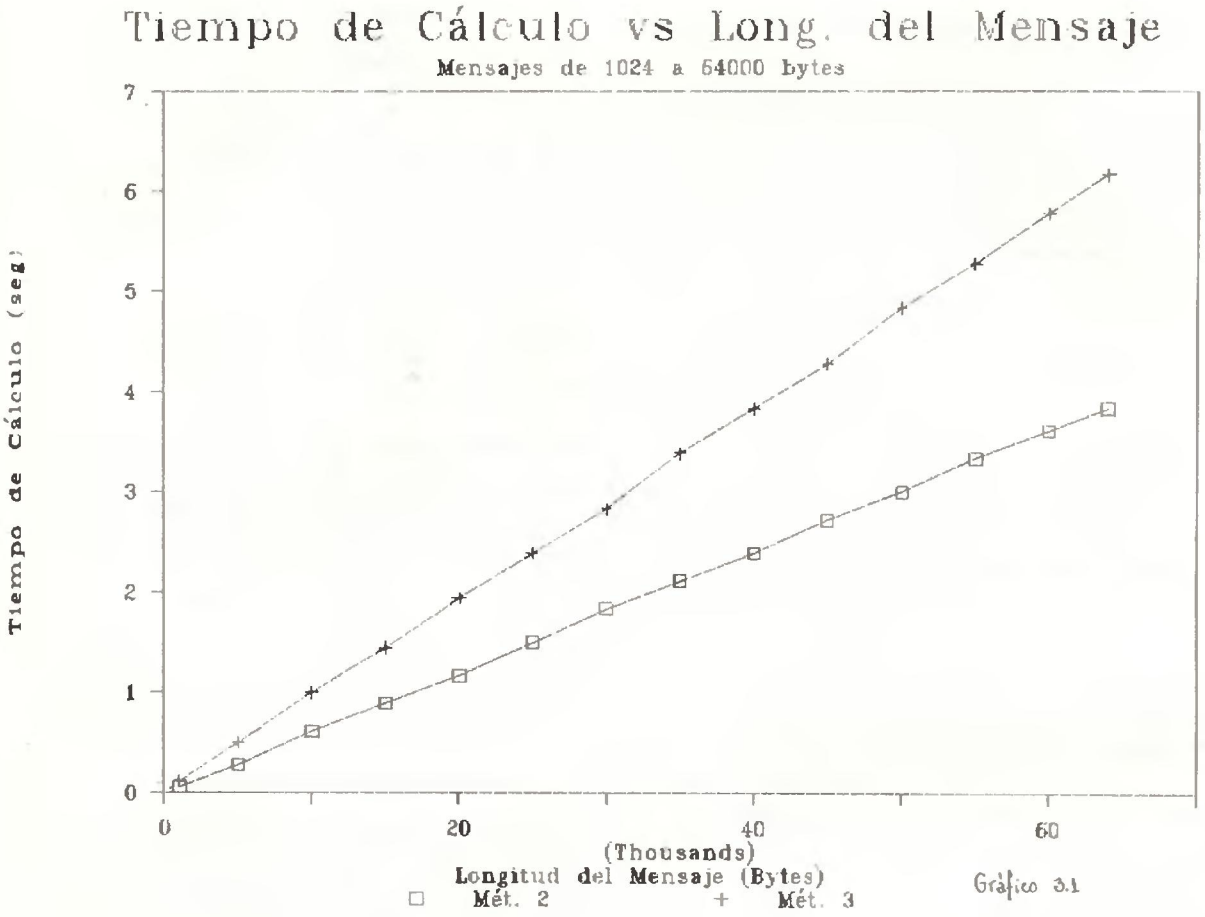


Gráfico 3.1

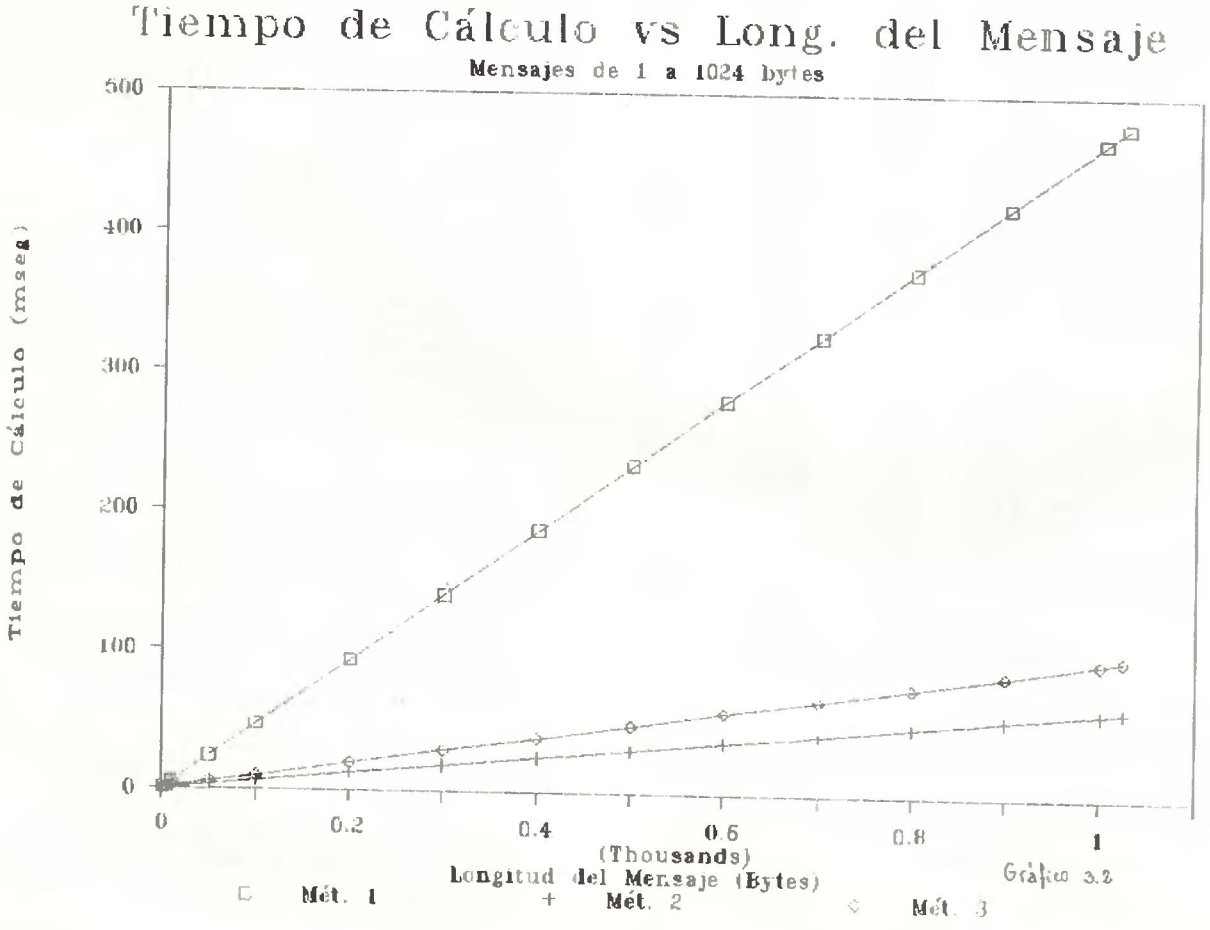
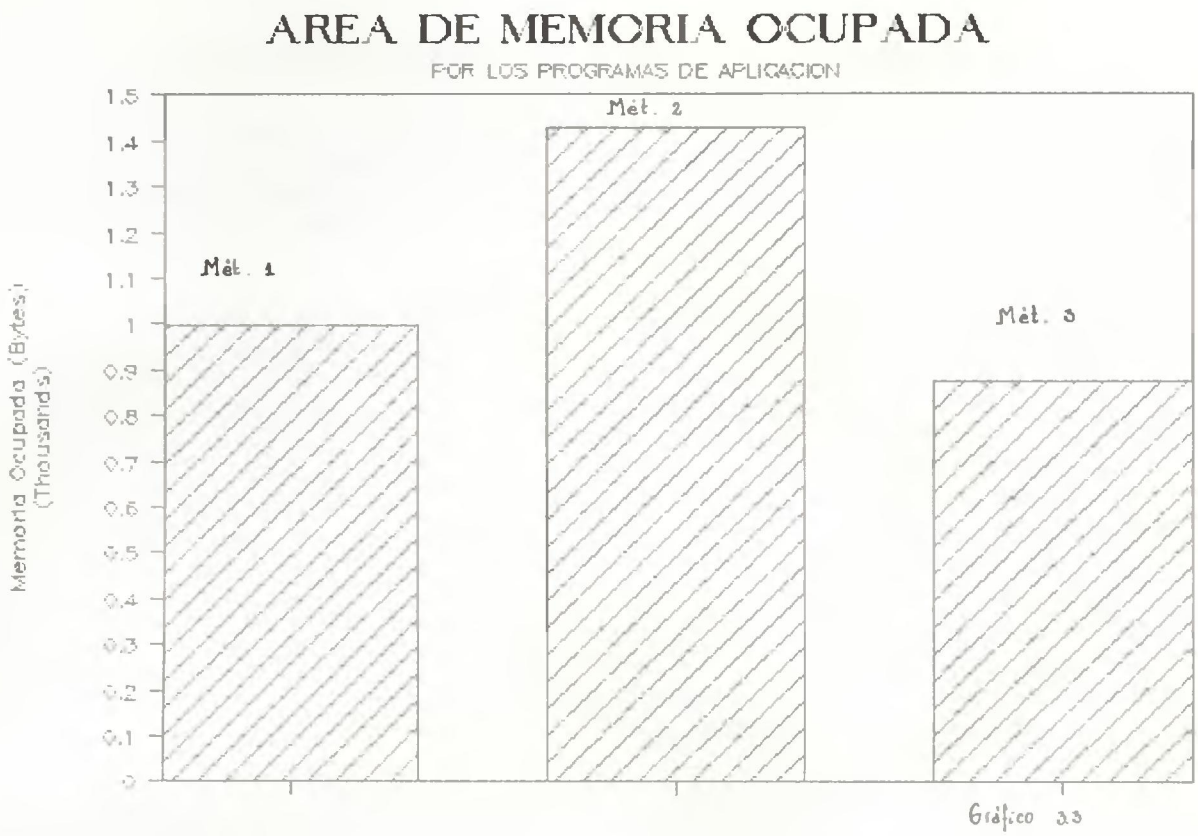


Gráfico 3.2



4 Conclusiones

1. El algoritmo de cálculo del CRC óptimo por su velocidad es aquel que toma el mensaje por bytes y utiliza búsqueda de los CRCs de un byte de longitud en una tabla.

2. Considerando además de la velocidad de ejecución el área de memoria que ocupan los programas de aplicación de los algoritmos, se establece que el cálculo "on the fly" es el apropiado porque por un lado es más rápido que realizando bit a bit, y por otro no se almacena la tabla de los CRCs de un byte.

BIBLIOGRAFIA

[1] Albertengo, Guido, Parallel CRC Generation, IEEE MICRO, Volumen 10, Número 5, Octubre 1990, Pgs. 63-71.

[2] Egas, Carlos, Codificadores y Decodificadores para Códigos de Bloque Lineales, E.P.N., 1987.

[3] Hamming, Richard, Coding and Information Theory, Prentice-Hall, New Jersey, 1980.

[4] IETEL, Códigos Detectores y Correctores de Errores, Marzo 1985.

[5] Perez, Aram, Byte-wise CRC Calculations, IEEE MICRO, Volumen 10, Número 5, Octubre 1990, Pgs. 63-71.

[6] Shouse, D.V., "On the Fly" CRC-16 Byte-wise CRC Calculation for 8088-based Computers, IEEE MICRO, Volumen 5, Número 2, Abril 1985, Pgs. 67-75.

[7] Shu Lin y Costello, Error Control Coding: Fundamentals and Aplications, Prentice-Hall, New Jersey, 1983.

BIOGRAFIAS

Medina Alicia, nació en Mira (Carchi), el 25 de Octubre de 1964.

Realizó sus estudios superiores en la Facultad de Ingeniería Eléctrica, especialización Electrónica y Telecomunicaciones de la E.P.N.. Actualmente desarrolla el presente tema como tesis de grado.

Montalvo Luis sus datos biográficos se encuentran en el articulo "Diseño y Construcción de Unidades Terminales Remotas (RTU) para Sistemas de Supervisión, Control y Adquisición de Datos (SCADA) publicado en este volumen.