4 Algoritmos de generación por *hardware*

4.0 Introducción

Si bien en el capítulo 2 se describieron una serie de técnicas para generar secuencias de números aleatorios mediante ordenador, en muchas aplicaciones usadas en telecomunicaciones es atractivo, e incluso imprescindible, la generación de este tipo de secuencias prescindiendo de éstos y utilizando dispositivos electrónicos de tipo común. Las especificaciones que se piden a las secuencias generadas son las mismas que en el caso anterior. Sin embargo, la eficiencia, la rapidez de obtención de los bits y la menor complejidad estructural de los dispositivos implicados adquieren una mayor importancia de la que tendrían en el caso de que las secuencias se obtuviesen a partir de un programa ejecutado sobre un ordenador.

En este capítulo se van a ver una serie de dispositivos que, con técnicas descritas en el capítulo 3 y otras nuevas, generarán secuencias pseudoaleatorias empleando para ello dispositivos electrónicos comúnmente utilizados y conocidos. Vamos a dividir estos generadores en tres grupos diferentes. Al primer grupo pertenecerán aquellos en los que la generación se basa en combinar generadores pseudoaleatorios básicos, como pueden ser los registros de desplazamiento con realimentación lineal (LFSR) mediante funciones combinadoras y añadiendo funciones no lineales según se requiera.

Al segundo grupo pertenecerán todos aquellos que se basan en el empleo de técnicas de control de reloj. La teoría básica de estos dos primeros grupos se vio en el capítulo 3. Finalmente habrá un tercer grupo de generadores que por su naturaleza no emplean ninguna de las técnicas de los anteriores y, por tanto, se verán como generadores especiales.

4.1 Generadores basados en funciones combinadoras

4.1.1 El generador de Jennings

Este es un esquema propuesto por S.M. Jennings en 1980 [JEN80] y que usa un multiplexor para combinar las secuencias de salida de dos registros de desplazamiento con realimentación lineal (LFSR-1 de *m* celdas y LFSR-2 de *n* celdas) tal como se muestra en la figura 4.1 [JEN82].

El generador produce la señal de salida $z(t)$, $t \geq 0$ de la siguiente forma: se fija un entero positivo h que cumpla que $h \leq \min(m, \log_2 n)$ y una función de transformación $0 \leq i_0 < i_1 < ... < i_{h-1} \leq m - 1$, que actúe sobre h celdas del LFSR-1 de forma que en cada momento, para $t \geq 0$, forme el número: $u(t) = a(t+i_0) + a(t+i_1)2 + ... + a(t+i_{h-1})2^{h-1}$ a partir de dichas celdas y lo transforme en el número $ß(u(t)) = s_0(t) + s_1(t)2 + ... + s_{k-1}(t)2^{k-1}$, donde $k = \lfloor log_2 n \rfloor$.

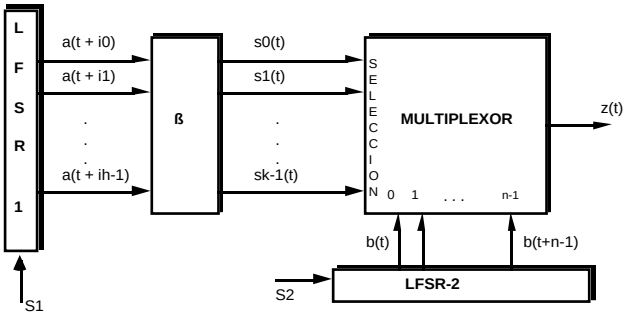

Fig. 4.1 Generador de Jennings

Realiza, pues, una aplicación inyectiva $ß : \{0, 1, ..., 2^h-1\} \rightarrow \{0,1, ..., n-1\}$. La familia de transformaciones ß posibles es enormemente grande. Con esto, la señal de salida del generador se define como:

$$z(t) = b[t + ß(u(t))]$$

Jennings [JEN80] demuestra que si ambos LFSR tienen polinomios primitivos y sus órdenes (número de celdas) son mutuamente primos entre sí (mcd(m,n) = 1), la secuencia de salida tendrá un periodo:

$$T = (2^m-1)(2^n-1)$$

y la complejidad lineal estará acotada superiormente por:

$$\Lambda(z) \leq n\left(1 + \sum_{i=1}^{h} C_i^m\right)$$

donde se dará la igualdad si las posiciones de las celdas del LFSR-1, que se eligen como entrada a la función de transformación ß, están equiespaciadas. La semilla de este generador la constituye el contenido inicial de las celdas de ambos LFSR.

Una mirada más profunda a la definición de la salida $z(t)$ revela que, independientemente de cuál sea la función de transformación ß, la señal se puede expresar como una combinación lineal de las $b(t)$, con los coeficientes dependiendo principalmente de S_1. Este será el talón de Aquiles si este

generador se emplea en aplicaciones como la criptografía, ya que se le puede aplicar un test de consistencia lineal (LCT) [ZEN89] para averiguar los parámetros del generador y poder determinar así completamente la secuencia de salida. Esto se verá en el capítulo 7. Si los polinomios de realimentación son conocidos, un posible atacante podría, conociendo tan sólo una porción de $N \geq m+n2^h$ bits de la secuencia $z(t)$ de salida, determinar completamente la estructura del generador. Para ello, bastaría que aplicase 2^{m+h} tests de consistencia y tan sólo le quedaría obtener la semilla S_1 utilizando una búsqueda exhaustiva (consistente en probar todas las posibilidades).

Este generador está propuesto por la EBU (European Broadcasting Union) para realizar el cifrado en el estándar de transmisión de televisión MAC. Una descripción de cómo se realiza el cifrado en este sistema se puede ver en el capítulo 7, mientras que en el último capítulo se realiza un análisis de este generador.

4.1.2 El generador de Geffe

Otra forma muy simple de combinar registros de desplazamiento realimentados linealmente mediante un multiplexor es la que muestra la figura 4.2 [GEF73] En esta estructura, las secuencias producidas por dos LFSR se usan como entradas de un multiplexor y, para introducir más aleatoriedad, se emplea la secuencia producida por otro multiplexor en la entrada de selección para decidir cuál de los dos LFSR de entrada es el que va a dar su bit a la salida. Esta estructura también se puede poner en la forma que muestra la figura 4.3.

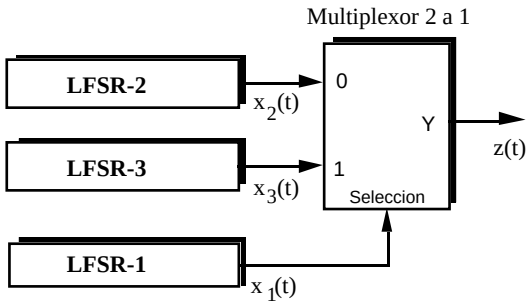

Fig. 4.2 Generador de Geffe

Si llamamos $x_2(t)$ y $x_3(t)$ a las señales binarias de entrada del multiplexor, y $x_1(t)$ a la señal que ataca a la entrada de selección, la salida del generador en el instante t es:

$$z(t) = x_1(t)x_3(t) \oplus \overline{x_1(t)}x_2(t) = x_2(t) \oplus x_1(t)(x_2(t) \oplus x_3(t))$$

o también:

$$z(t) = x_3(t) \oplus \overline{x_1(t)}(x_2(t) \oplus x_3(t))$$

Si los registros de desplazamiento con realimentación lineal LFSR-1, LFSR-2 y LFSR-3 tienen grados n_1, n_2 y n_3 respectivamente, y polinomio de realimentación primitivo, el periodo de la secuencia de salida z(t) de este generador será:

$$T = mcm(2^{n_1} - 1, 2^{n_2} - 1, 2^{n_3} - 1)$$

y la complejidad lineal:

$$\Lambda = n_3 n_1 + (n_1 + 1)n_2$$

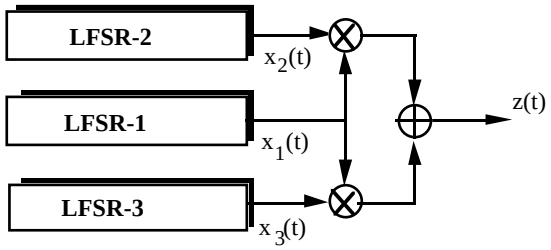

Fig. 4.3 Otra forma del generador de Geffe

Este generador no es aconsejable en ciertas aplicaciones como la criptografía, ya que presenta una debilidad debida a que existe una probabilidad de coincidencia entre la salida z(t) y la secuencia de entrada x_2(t) que es mayor que 1/2:

$$P = Prob(z(t) = x_2(t)) = Prob(x_1(t) = 0) + Prob(x_1(t) = 1)\, Prob(x_3(t) = x_2(t)) = \frac{1}{2} + \frac{1}{4}$$

y lo mismo ocurre entre z(t) y x_3(t) ya que

$$P = Prob(z(t) = x_3(t)) = \frac{1}{2} + \frac{1}{4}$$

y se puede estimar de igual modo que la anterior. Además, si un posible atacante conociera los polinomios de realimentación de los registros de desplazamiento y éstos fueran primitivos y de grado menor que un cierto valor n, podría romper fácilmente el generador empleando para ello un ataque basado en el método del síndrome lineal [ZEN89], y podría obtener el resto de la estructura del generador necesitando tan sólo una porción de la secuencia de salida de longitud $N = 37n$.

Para llevar a cabo este ataque tan sólo necesitaría realizar $896n$ operaciones binarias. Existen además otras técnicas basadas en los ataques por correlación que permiten criptoanalizar sistemas de cifrado en flujo basados en esta estructura. Tanto el método del síndrome lineal como los ataques por correlación para secuencias empleadas en sistemas criptográficos se tratarán con más detalle en el capítulo 7.

4.1.3 El generador de umbral

Este generador de secuencias pseudoaleatorias, propuesto por J.O Bruer en 1982 [BRU82], tiene una estructura basada en N registros de desplazamiento con realimentación lineal obtenida mediante polinomios primitivos combinados de la siguiente forma:

$$z_i = \begin{cases} 1 \quad si \quad \sum_{j=1}^{N} x_{ji} > N/2 \\ 0 \quad en \quad otro \quad caso \end{cases}$$

En cada instante t, la salida de los N registros de desplazamiento se suman, y el resultado de esta suma se pasa a través de un detector de umbral, tal como se ve en la figura 4.4. La salida del generador $z(t)$ será igual a 1 si el número de unos a la entrada excede de $N/2$, y 0 en caso contrario.

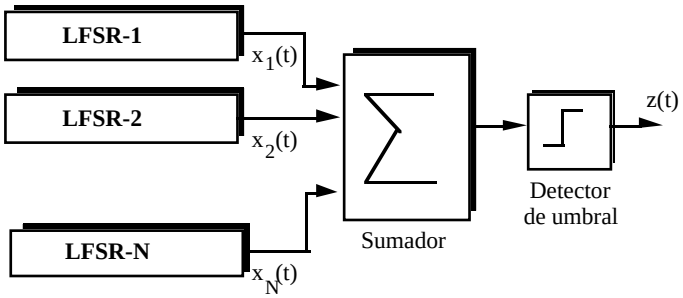

Fig. 4.4 Generador de umbral

La salida de este generador estará balanceada sólo si el número de LFSRs utilizado es impar. Veamos un ejemplo con $N = 3$ en la que la salida equivalente, puesta en notación algebraica normal (ANF) sea: $z_i = x_{1i}x_{2i} \oplus x_{1i}x_{3i} \oplus x_{2i}x_{3i}$.

Esta forma de representación viene determinada por la suma módulo 2 de todos los productos de segundo orden. Si el número de celdas de los 3 LFSRs son L_1, L_2 y L_3 respectivamente y sus periodos P_1, P_2 y P_3, el periodo P de la secuencia $z(t)$ y la complejidad lineal $\Lambda(z)$ serán:

$$P = P_1 \cdot P_2 \cdot P_3 \qquad \Lambda(z) = L_1L_2 + L_1L_3 + L_2L_3$$

Hay que tener en cuenta que existirá una cierta correlación entre la secuencia de salida $y(t)$ y cada una de las secuencias producidas por los LFSRs. Esto representa un inconveniente si se usa este generador para aplicaciones de cifrado en flujo, ya que puede ser aprovechada para criptoanalizar el sistema. Este generador no es aconsejable para aplicaciones criptográficas de cifrado en flujo, ya que presenta ciertas debilidades frente a ataques basados en la correlación.

4.1.4 El generador de Pless

Veamos en este apartado un esquema propuesto por V.S. Pless [PLE77] basado en usar básculas *J-K* como funciones combinadoras de secuencias generadas por registros de desplazamiento con realimentación lineal. Este tipo de estructura preserva las buenas propiedades de aleatoriedad de las secuencias producidas por los LFSR y, además, soluciona el problema de la linealidad inherente en estos dispositivos, que puede ser importante en ciertas aplicaciones como las criptográficas debido a la existencia de numerosos ataques que aprovechan la debilidad de su linealidad inherente.

La báscula *J-K* es bien conocida por su utilidad en circuitos digitales (ya estudiado en el capítulo anterior). Es un dispositivo de dos entradas (*J*,*K*) y dos salidas (donde una salida es la complementaria de la otra), y que opera de la siguiente forma: Consideremos un par ordenado (*J*,*K*) para representar la entrada. Un (0,0) deja la salida sin alterar, es decir, con el valor que tenía antes de cambiar la entrada. Un (1,1) a la entrada invierte el estado que tendría la salida en el instante anterior (de 0 a 1 o viceversa). Una entrada (0,1) produce un 0 a la salida y, finalmente, una entrada (1,0) produce un 1 a la salida.

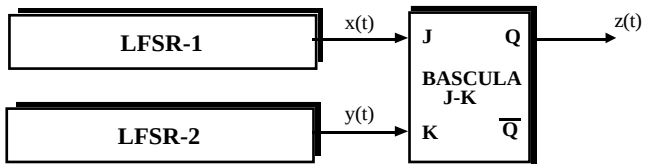

Fig. 4.5 Generador de Pless

Veamos una primera estructura basada en una sola báscula *J-K* que combina la salida de los LFSRs, tal como muestra la figura 4.5. De esta estructura se pueden hacer las siguientes afirmaciones:

Si el LFSR-1 produce una secuencia de periodo $P_1 \neq 1$, y el LFSR-2 produce una secuencia de periodo $P_2 \neq 1$, y se cumple que el mcd(P_1,P_2)=1, y que tanto P_1 como P_2 son impares, entonces el periodo de la secuencia que produce esta estructura P es $P = P_1P_2$. Esta estructura es muy fácil de implementar debido a la sencillez de todos los elementos que la componen y, además, presenta buenas propiedades en el caso de errores en la transmisión de la secuencia producida, ya que si este error se produce en un bit producido por el *flip-flop J-K*, se tratará de un único error que no afectará a los otros bits. Si se produce un error en el estado interno de la báscula *J-K*, éste afectará a todos los bits mientras la entrada (*J*,*K*) sea (0,0) o (1,1). Sin embargo, este error se corregirá tan pronto como la entrada pase a valer (1,0) o (0,1). Por tanto, tendremos un flujo de bits completamente incorrecto pero muy fácil de detectar o un flujo de bits completamnte correcto.

Sin embargo, esta estructura no preserva las buenas propiedades estadísticas que presentan las secuencias producidas por los LFSR que actúan de entrada en la báscula *J-K*. Si en la entrada de la bascula hay un 1, la probabilidad de que la siguiente salida sea un uno es menor que 1/2, y ocurre de forma similar cuando la salida es un cero.

Esto también es un problema si se pretende emplear esta estructura en aplicaciones de cifrado, ya que el sistema será más susceptible a ataques estadísticos que una secuencia realmente aleatoria.

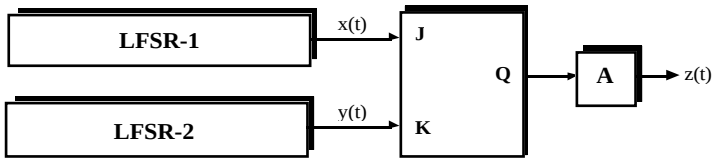

Fig. 4.6 Mejora sobre el generador de Pless

En el capítulo 3 se vio que la estructura de un LFSR de L celdas se puede determinar analizando $2L$ bits producidos a la salida. Este problema se mantiene con esta estructura, ya que si el LFSR-1 tiene m celdas y el LFSR-2 tiene n celdas, se puede determinar la estructura estudiando tan sólo $2n + 2m$ bits producidos por ella. El problema de la no aleatoriedad de la salida se puede paliar, en parte, añadiendo a la salida de la estructura anterior un dispositivo A que elimine bits de forma alternada, tal como muestra la figura 4.6. Para esta estructura, también se cumple que si el LFSR-1 tiene periodo impar P_1, y el LFSR-2 tiene periodo impar P_2, y el mcd(P_1,P_2) = 1, el periodo de la secuencia de salida será $P = P_1P_2$. Además, el alternador A restaura parte de las buenas propiedades estadísticas de los LFSRs de entrada, ya que 2 es primo relativo con P_1P_2 cuando P_1P_2 es impar. La desventaja de esta estructura es que emite un bit cada dos pulsos de reloj, por lo que el registro de desplazamiento debe operar al doble de la velocidad de la secuencia de entrada.

Tabla 4.1

LFSR	*Nº celdas*	*Periodo*	*Nº de elecciones*	*Factorización de P*
1	5	31	6	31
2	19	524287	27594	524287
3	7	127	18	127
4	17	131071	7710	131071
5	9	511	48	7*73
6	16	65635	2048	3*5*17*257
7	11	2047	176	23*89
8	13	8191	630	8191

Otra posible estructura es la que se muestra en la figura 4.7, donde se emplean 4 LFSRs, 3 básculas J-K y 3 alternadores. Si se asume que r_1, r_2, r_3 y r_4 son las celdas de los 4 LFSR respectivamente, para poder obtener los parámetros de la estructura se necesitarían al menos $2^{(2^{r_1}+r_2+2^{r_3}+r_4)}$ bits alternados. Este ya es un valor elevado para valores de r_1, r_2, r_3 y r_4 pares y moderadamente grandes. Esta estructura presenta cierto atractivo para usarla en aplicaciones de tipo criptográfico. Las desventajas de esta estructura son, que es difícil de simular mediante un ordenador, y no es posible ejecutarla en tiempo real. Como ventaja tiene que es muy fácil de implementar.

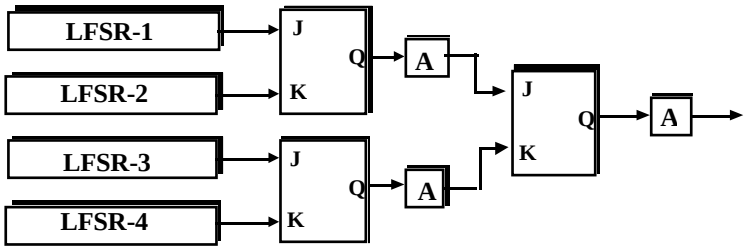

Fig. 4.7

Una última estructura presentada por Pless es la de la figura 4.8, que mantiene las propiedad de ser difícil de romper mediante cálculos lineales, que ya presentaba la estructura anterior, además de permitir su ejecución en tiempo real. El dispositivo con cajas numeradas de 1 a 4 es un contador cíclico y transmite el contenido de la caja $i + 1 \pmod 4$ a la derecha después de que se haya transmitido el contenido de la caja i. En esta estructura, si se usan M LFSR de forma que las longitudes de sus secuencias de salida sean impares y mutuamente primas dos a dos, el periodo P de la secuencia de salida será:

$$P = \sum_{i=1}^{M} P_i$$

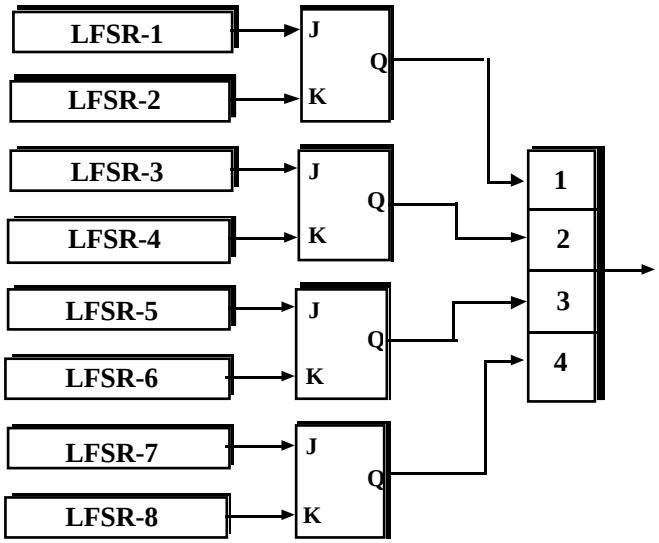

Fig. 4.8

La tabla 4.1 muestra una posible elección del número de celdas (r_i) para los $N = 8$ LFSR. Además, se eligen polinomios primitivos para todos ellos. De dicha tabla se puede observar que los

periodos de los 8 LFSR son impares y mutuamente primos dos a dos, por lo que el periodo de la estructura completa será su producto, que es mayor que 10^{28}.

El número de elecciones diferentes de polinomios primitivos que sean capaces de producir esos periodos para los 8 LFSRs se muestra en la cuarta columna de la tabla.

Para la estructura completa, el número de elecciones diferentes posibles de los 8 polinomios primitivos para los 8 LFSR será el producto de los valores de la quinta columna y será mayor que 2×10^{20}. Esto implica un número de posibilidades enorme que impediría, en el caso de que este generador se empleara, por ejemplo, en un sistema criptográfico, romper el criptosistema mediante el intento simple de todas las posibles combinaciones por parte de un usuario no autorizado.

Aún se podría hacer una variación en este último esquema que consistiría en la sustitución de los LFSR por contadores módulo *n* (formados por la interconexión de básculas *J-K*). Estos contadores podrían ser a su vez lineales o no. Estos últimos son más difíciles de determinar incluso que los LFSR, en el caso de un criptoanálisis contra un sistema de cifrado en flujo basado en esta estructura.

4.1.5 El generador de Rueppel

Este generador consiste en *N* registros de desplazamiento con realimentación lineal y un subsistema combinador no lineal consistente en un sumador de reales y memoria de longitud $\log_2 N + 1$, tal como muestra la figura 4.9. Fue propuesto por R.A. Rueppel [RUE86] en un intento de evitar el compromiso entre inmunidad a la correlación y la complejidad lineal que aparece en las funciones combinadoras sin memoria que se expuso en el capítulo 3. Este generador consiste en un subsistema conductor compuesto de *N* registros de desplazamiento con realimentación lineal y una función combinadora no lineal que incluye un sumador de reales y una memoria.

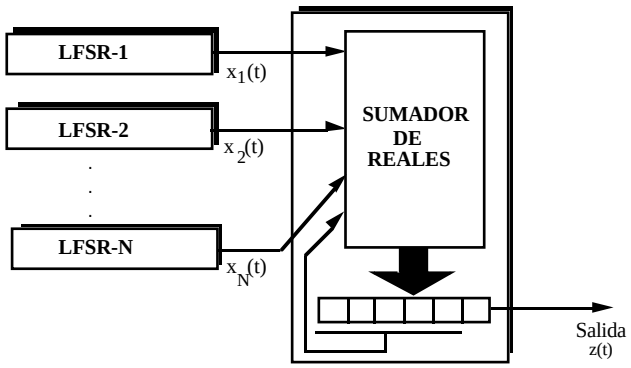

Fig. 4.9 Generador de Ruepple

El funcionamiento es el siguiente: En el instante 1, se suman como números reales (y no módulo 2) el primer bit de cada una de las *m*-secuencias producidas por los *N* registros de desplazamiento $x_i(1)$ (i = 1,..., N). El resultado es un número de log$N$ + 1 bits que se almacena en una

memoria interna. Una vez hecho esto, se desplaza el contenido de esta memoria un bit a la derecha, con lo que se obtiene el primer bit de la secuencia de salida a partir del bit menos significativo. En el instante 2, el contenido de la memoria y la nueva salida de los N LFSRs $x_i(2)$ (i = 1, ..., N), se vuelven a sumar (suma real) iterando así todo este procedimiento.

Si se eligen los polinomios de realimentación de los LFSR primitivos (entonces producen m-secuencias de periodo $2^{L_i} - 1$, donde L_i es el número de celdas y $1 \le i \le N$) y se impone la restricción de que los grados (número de celdas) sean relativamente primos uno a uno ($\forall i \neq j \quad 1 \le i,j \le N$, mcd($L_i$, L_j) = 1), la secuencia que produce el generador presenta las siguientes condiciones:

1) Gran periodo (el máximo): $P = \prod_{i=1}^{N} (2^{L_i} - 1)$.
2) Complejidad lineal aproximadamente la máxima:
$$\Lambda \approx \Lambda_{max} = \prod_{i=1}^{N} (2^{L_i} - 1)$$
3) Excelente propiedades estadísticas (los unos y los ceros son casi equiprobables).
4) Es posible conseguir inmunidad a la correlación de orden N - 1 (la máxima) debido a la memoria interna.

La inmunidad a la correlación, tal como se vio en el capítulo 3, define el grado de incorrelación (independencia estadística) entre las secuencias que producen los LFSR (subgeneradores) y la secuencia de salida. Si bien en ciertos casos esta propiedad no es relevante, en otros como la criptografía es de vital importancia cuando se emplean estructuras como ésta, basadas en varios subgeneradores, y una función no lineal que los combina. Si las secuencias no están incorreladas, un posible atacante puede aprovechar esta información mutua para realizar un ataque que permita obtener los parámetros de la estructura, incluída la clave secreta (que es el contenido inicial de las celdas de los N registros de desplazamiento).

Tal como vimos en el capítulo 3, el hecho de introducir memoria en la función combinadora no lineal permite conseguir simultáneamente máxima complejidad lineal y máxima inmunidad a la correlación.

4.1.6 Generador de Tatebayashi

En 1989, Tatebayashi, Matsuzaki y Newman [TAT89] propusieron una modificación sobre el generador de Ruepple (visto en el apartado anterior) para aplicaciones de cifrado en comunicaciones móviles. La estructura resultante se muestra en la figura 4.10, donde se ve que la modificación consiste en añadir un registro de control de longitud N (que llamamos REGCTRL). Con ello se consigue producir diferentes secuencias en lugar de una sola al usar diferentes vectores iniciales, por lo que se puede usar como clave durante una comunicación. Conviene recordar que al variar el contenido inicial de las celdas de los subgeneradores (registros de desplazamiento) lo que se consigue es modificar el punto en que se inicia la secuencia, pero ésta es siempre la misma. Al modificar el valor del registro REGCTRL se pueden conseguir 2^N secuencias diferentes.

Su contenido debe ser constante durante una transmisión y se puede considerar como un aumento de la clave secreta que hasta ahora estaba formada por el estado inicial de las celdas de los LFSR.

El funcionamiento de este generador es el siguiente: El REGCTRL tiene N bits de forma que el i-ésimo bit se suma módulo 2 (suma OR exclusiva) con la secuencia de salida $\{x_i\}$ del i-ésimo LFSR bit a bit, con lo que resulta la secuencia $\{y_i\}$. En el módulo sumador, las secuencias $\{y_i\}$ se suman como números reales, al igual que ocurría en el generador de Rueppel.

La adición de este registro de control no modifica las propiedades de periodo, complejidad y estadística que mostraba el generador de Rueppel.

También para este generador se eligen N LFSRs con grados mutuamente primos entre sí y polinomios de realimentación primitivos. Con estas condiciones, la secuencia de salida $z(t)$ tendrá las siguientes propiedades:

1) Periodo de salida máximo: $P = \prod_{i=1}^{N} (2^{L_i} - 1)$
2) Complejidad lineal casi máxima $\Lambda(z) \approx P_{max}$
3) Los unos y los ceros están balanceados.
4) Inmunidad a la correlación de máximo orden N - 1.

Se puede comprobar fácilmente la propiedad 3 mediante el siguiente razonamiento: Puesto que todos los LFSR ($LFSR_i$) del subsistema conductor generan m-secuencias $\{x_i\}$, ya se ha dicho que sus ceros y unos están balanceados. Las secuencias $\{y_i\}$ obtenidas también tendrán los unos y ceros balanceados, puesto que dichas secuencias son, o bien iguales a $\{x_i\}$, o bien iguales a $\{x_i\}$ pero complementadas.

Llamemos a $u_k(j)$ el bit menos significativo de la suma real de los N bits obtenidos de las secuencias $\{y_i\}$ (i = 1,...,N) más el acarreo (excepto el bit de la secuencia y_k) en el instante j:

$$u_k(j) = \sum_{i=1}^{N} y_i(j) + c_i(j)$$

El siguiente bit de la secuencia de salida del generador $z(t)$ es el bit menos significativo de la suma de todos los y_i y el acarreo $z(j) = y_k(j)+u_k(j)$. Entonces:

$$P(z=1) = P(y_k=1)P(u_k=0)+P(u_k=0)P(u_k=1) = 1/2P(u_k=0)+1/2P(u_k=1) = 1/2$$
$$P(z=0) = P(y_k=0)P(u_k=0)+P(y_k=1)P(u_k=1) = 1/2P(u_k=0)+1/2P(u_k=1) = 1/2$$

lo que demuestra el balanceo que existe entre los unos y los ceros en la secuencia $z(t)$ de salida.

También se puede demostrar el hecho de que el generador presente inmunidad a la correlación de máximo orden N - 1. Para ello, se define $u_k(j)$ de la misma forma que antes.

Puesto que en cada instante j, el nuevo bit de la secuencia de salida z(t) se obtiene mediante $z(j) = y_k(j) + u_k(j)$, entonces se cumple que $P(z=u_k) = P(y_k=0) = 1/2$ y que $P(z \neq u_k) = P(y_k=1) = 1/2$. Por tanto, la salida $z(t)$ estará incorrelada con $u_k(t)$, lo que implica que $z(t)$ presentará inmunidad a la correlación de orden N - 1, ya que u_k tiene N - 1 variables.

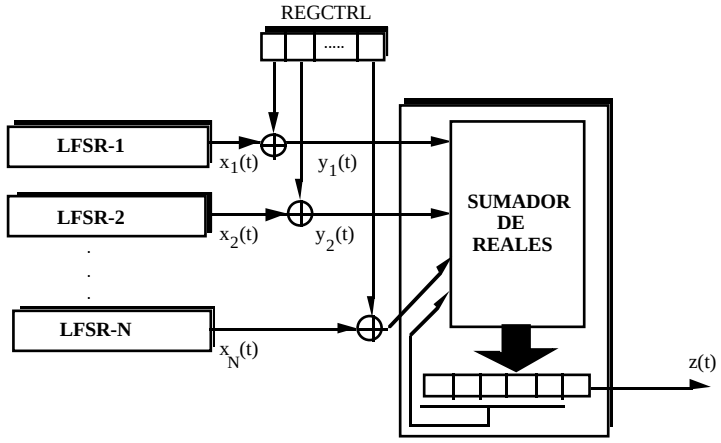

Fig. 4.10 Generador de Tatebayashi

4.2 Generadores basados en técnicas de control de reloj

Estos generadores se basan en controlar el reloj de un registro de desplazamiento con realimentación lineal (LFSR) mediante la salida de otro LFSR. Antes de ver estos generadores, recordemos algunos conceptos sobre técnicas de control de reloj [CHA88] vistos anteriormente en el capítulo 3. Consideremos un LFSR de control y un LFSR cuyo reloj es controlado por la salida del anterior, de polinomio de realimentación irreducible f(x) de grado n y orden N, y cuya salida llamamos $u(t)$. El funcionamiento de la técnica de control de reloj es el siguiente: Si después de la salida (t-1)-ésima del sistema, el LFSR de control da un entero no negativo b_t, se deberán dar b_t pasos de reloj al LFSR controlado antes de que éste dé su siguiente bit de salida u(t). Además, hay que dar un paso al LFSR de control para reiterar el proceso anterior. Si llamamos $s\{t\}$ a la salida del LFSR controlado, si no se aplica sobre éste la técnica de control de reloj, su salida $u\{t\}$ cuando sí se aplique esta será:

$$u(t) = s\left(\sum_{k=0}^{t} b_k\right) \qquad \text{para } t \geq 0$$

si la salida inicial es $u(0) = s(b_0)$. Además, se debe tener en cuenta que el LFSR de control tendrá una salida periódica de periodo M, con lo que $b_{t+M} = b_t$ para todo $t \geq 0$. Entonces se cumplirá que:

$$u(i + jM) = s(jp + c(i)) \qquad 0 \leq i < M \qquad j \geq 0$$

donde:

$$c(t) = \sum_{k=0}^{t} b_k \quad (1) \qquad \text{y} \qquad p = \sum_{k=0}^{M-1} b_k \quad (2)$$

ya que $c(t + M) = c(t) + p$ para todo $t \geq 0$. Además, se debe imponer la condición de que p y N sean mutuamente primos entre sí, y sea N el orden del polinomio de realimentación f(x).

Un caso especial de gran importacia es cuando b_t toma solamente los valores 0 y 1, dependiendo de cuál sea el bit de salida del LFSR de control. El generador resultante se llama de marcha y espera (*stop & go*), y se verá con más detalle en el apartado 4.2.4. Otro caso es cuando el registro de control no es único, sino que es una cascada de generadores (es el caso del generador de Gollmann que se estudiará en el apartado 4.2.4 B).

A continuación se verán algunos generadores de secuencias pseudoaleatorias basados en esta técnica de control de reloj.

4.2.1 Generadores de control de reloj con modulación de fase

En una descripción de control de reloj como la del apartado anterior, los valores de b_t deberían ser bastante pequeños, ya que si no el LFSR controlado necesitará funcionar a una velocidad mucho mayor que la tasa a la que produciría bits de salida. Para solucionar esto, se puede modular en fase la salida $u(t)$ para producir una salida $u'(t) = u(t + \lambda_t)$ de forma que se obtenga el mismo efecto que cuando se tienen b_t grandes, donde $\{\lambda_t\}$ es una secuencia de enteros de periodo M [CHA88]. A partir de aquí, se pueden obtener unas ecuaciones similares a las del apartado anterior. Para este caso de modulación de fase se cumplirá que $u'(t) = s(c'(t))$, donde $c'(t) = c(t + \lambda t)$ y además $c'(t)$ cumplirá la condición $c'(t + M) = c'(t) + p$.

Al igual que en el apartado anterior, también se puede definir una b'_t como $b'_t = c'(t) - c'(t-1)$ y que satisface la condición $b'_{t+M} = b'_t$.

El LFSR de control y la secuencia $\{\lambda_t\}$ deben tener periodos diferentes, por ejemplo M' y M'' respectivamente. Entonces podemos definir el periodo de control global como el menor múltiplo común entre M' y M'' y el valor p visto anteriormente vendrá dado en este caso por una suma sobre M'/M'' periodos del LFSR controlado, y será, por tanto, un múltiplo de $M'M'$. Por tanto, para mantener la condición de que p sea mutuamente primo con N (donde N es el orden del polinomio irreducible del LFSR controlado), se debe cumplir que M/M' sea mutuamente primo con N. Veamos algunos generadores de secuencias pseudoaleatorias que se basan en esta técnica.

A) El generador mezclador de modulación de fase de MacLaren-Marsaglia

Como ejemplo de este tipo de generadores se encuentra el llamado mezclador de MacLaren-Marsaglia [MAC65], cuya estructura muestra la figura 4.11, y que usa un generador pseudoaleatorio auxiliar para producir una secuencia periódica de enteros $\{x_t\}$ en el rango, por ejemplo, de 0 a K - 1.

Estos enteros se usan para seleccionar una dirección en una memoria de acceso aleatorio (RAM) de K direcciones, cada una de ellas almacena un bit. El bit que contiene la posición determinada por x_t se lee para producir el nuevo bit de salida $u'(t)$, y en su lugar se escribe el valor que contiene $u(t)$.

Los valores de K elevados no son recomendables, no sólo por el hecho de que se necesitará una RAM de elevada capacidad (y por tanto elevado coste), sino también por el hecho de que este generador necesita una inicialización que concluirá cuando todas las posiciones de la RAM se hayan llenado.

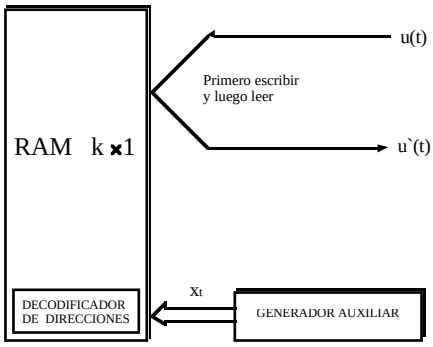

Fig. 4.11 Generador de MacLaren-Marsaglia

Un ejemplo práctico de este generador consistiría en usar este sistema para mezclar la salida de un LFSR de longitud n con un polinomio primitivo de realimentación de orden $N = 2^n-1$ controlado mediante una técnica de control de reloj. Esta, a su vez, puede consistir en emplear otro LFSR de longitud n y periodo $M' = N$ cuya salida ataque a la entrada de reloj del anterior. Si elegimos el periodo del mezclador M'' de forma que sea primo y además mutuamente primo con N, el periodo global del control será $M = N \times M''$, y la complejidad lineal de la secuencia de salida estará acotada inferiormente por $N \times n$.

Si bien el empleo de esta técnica no mejora la complejidad final respecto al LFSR con entrada de reloj contralada por otro LFSR, sí mejora, sin embargo, las propiedades estadísticas de la secuencia pseudoaleatoria resultante.

B) Generador de producto escalar con el vector estado

Este es otro tipo de generador que realiza una modulación de fase de forma más potente que el anterior, y se basa en el producto escalar binario [LID86]. El producto escalar binario de dos vectores es el producto escalar reducido módulo 2. En la figura 4.12 se muestra cómo implementar este generador. En ella, hay un LFSR de control de periodo M' cuya salida ataca a la entrada de reloj de otro LFSR de n celdas con polinomio de realimentación irreducible f(x). Este LFSR controlado producirá una secuencia de salida $S_t = S(c(t)) = (s_0(t), s_1(t),..., s_{n-1}(t))$ en el instante t según las pautas que se han visto anteriormente ($c(t)$ está definida en la ecuación 1 del apartado 4.2), que se denominará el vector de estado. En esta misma estructura existe un generador pseudoaleatorio auxiliar que elige direcciones en una memoria de sólo lectura (ROM) que contiene n vectores. La salida de este subsistema constituído por el generador auxiliar y la ROM la denotamos por $U_t = \{U_t\}$, y tendrá un cierto periodo M''. La salida $W(t)$ de toda esta estructura en el instante t será el producto escalar de las secuencias S_t y U_t. El periodo de control global será igual al menor múltiplo común entre M' y M'', por lo que será además el periodo de la secuencia de salida $W(t)$. Tendremos, pues, que $W(i + jM) = U_i \cdot S(c(i) + jp)$ con $0 \le i < M$ y $j \ge 0$, donde el valor de p viene dado en la ecuación 2 del apartado 4.2. Para un valor i fijo, $\{W(i + jM)\}$ con j = 0, 1, 2,... será una p-decimación de la secuencia producida por el LFSR, cuyo reloj es controlado. $W(t)$ consistirá, pues, en M secuencias mezcladas.

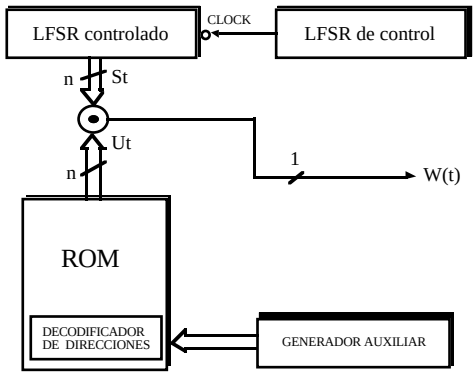

Fig. 4.12 Generador de producto escalar con el vector estado

La ventaja de este generador con respecto al del anterior apartado es que el rango de valores de $\lambda(U_i)$ puede llegar a ser del orden de 2^n-1.

Una variación de este generador podría obtenerse al sustituir el LFSR auxiliar y la memoria ROM por un LFSR de longitud mayor que *n*, del cual *n* bits se tomarían para formar la secuencia de vectores $\{U_t\}$. También se podría utilizar otro tipo de función combinadora no lineal diferente al producto escalar.

C) Generador en cascada

Se podrían conseguir mayores complejidades lineales que en las estructuras anteriores si el periodo *M* del LFSR que realiza el control del reloj fuera una potencia (N^r) del periodo *N* del LFSR cuyo reloj es controlado.

En este caso, el periodo de la secuencia de salida del conjunto sería $P = N^{r+1}$. Esto sugiere que podría ser interesante emplear estructuras en cascada de LFSRs, todas ellas de periodo *N*, y cuyo reloj esté controlado por el anterior.

Gollmann [GOL84] considera una estructura de este tipo que se describe en el apartado 4.2.5.B, en la que el control de reloj entre los sucesivos LFSR de la cascada se realiza mediante un solo bit. En este apartado se describe, sin embargo, un generador en el que el control entre las diferentes etapas de la cascada se realiza mediante varios bits en paralelo.

Para ello se construye una cascada de *L* etapas (excepto la primera y la última) como la de la figura 4.13. La entrada a cada una de estas etapas (a_t) consistirá en una secuencia de enteros de *k* bits. Uno de estos *k* bits se debe retrasar un paso, y se usará para elegir si al LFSR de *n* celdas y polinomio de realimentación primitivo se le deben dar uno o dos pulsos de reloj. Esto quiere decir que el LFSR se desplazará después de que se haya usado su salida b_t. Esta salida consistirá en *k* bits correspondientes a *k* celdas del LFSR ($k \leq n$), y éstos se deben sumar (módulo dos) bit a bit con los *k* bits de a_t para dar el vector suma c_t de *k* bits, que será además la entrada a una caja-*S* invertible que realizará una permutación de los enteros entre 0 a 2^k-1. Invertible significa que la permutación que debe realizar se hace mediante la transformación correspondiente, uno a uno, entre los enteros del rango considerado.

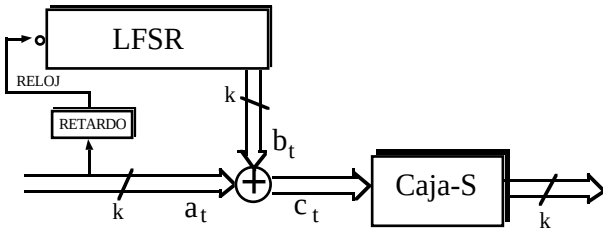

Fig. 4.13 Generador en cascada

La primera de las *L* etapas de este generador será un LFSR de *n* celdas con pulsos de reloj a intervalos regulares (por tanto no controlado) seguido de una caja-*S* invertible. La última etapa será como la de la figura 4.13, pero omitiendo la caja-*S*.

El periodo *P* de las *k* secuencias producidas por este generador será $P = N^L$, y todas ellas tendrán una complejidad lineal de al menos N^{L-1}. Si no se omite la caja-*S* de la última de las *L* etapas, la complejidad lineal de las *k* secuencias de salida aún puede incrementarse más.

Las cajas-*S* pueden ser diferentes para las distintas etapas, así como los polinomios primitivos de realimentación, pero siempre que el número de celdas de los LFSRs sea *n*.

D) El generador multiplicador de tasa binaria (BRM)

Este generador de secuencias pseudoaleatorias, propuesto por W.G. Chambers y S.M. Jennings en 1984 [CHA84] presenta la estructura de la figura 4.14.

Sea el LFSR-1 un registro de desplazamiento de *m* etapas que genera una secuencia de periodo $M = 2^m-1$ y el LFSR-2 un registro de desplazamiento de *n* etapas que genera una secuencia de periodo $N = 2^n-1$.

El generador BRM funciona de la siguiente forma: En el instante *t*, se da un pulso de reloj a ambos registros de desplazamiento. Entonces se hace que el LFSR-2 reciba a_t pulsos de reloj más, si a_t es el número que se obtiene de examinar el contenido de *k* celdas del LFSR-1. Para ello se debe asumir que $k < m$, de forma que $0 \leq a_t \leq 2^k-1$. La salida *z(t)* del sistema será la salida del LFSR-2 después de que ambos registros de deplazamiento hayan hecho un desplazamiento, y se haya desplazado al LFSR-2 a_t veces más. Si llamamos $\{s(t)\}$ a la secuencia generada por el LFSR-2, y $\{z(t)\}$ a la secuencia de salida del sistema, podemos representar al sistema para $t = 0, 1, 2,...$ como:

$$z(t) = s\left(t + \sum_{i=0}^{t} a_i\right)$$

puesto que siempre se cumple que $a_{t+M} = a_t$ para todo instante de tiempo *t*, para $0 \leq i \leq M-1$ y para todo *j* tendremos que $z(i + jM) = s(jp + d_i)$ donde:

$$d_i = i + \sum_{l=0}^{i} a_l \qquad \text{y} \qquad p = M + \sum_{l=0}^{M-1} a_l$$

Durante un ciclo completo de *M* pulsos de reloj producidos por el LFSR-1, se producirán cada una de las *m*-tuplas binaria distintas de cero como estados del LFSR-1.

Esto implica que cada *k*-tupla (excepto la de *k* ceros) entrará en el bloque BRM 2^{m-k} veces, por tanto:

$$\sum_{l=0}^{M-1} a_i = 2^{m-k} \sum_{i=1}^{2^k-1} i = 2^{m-1}(2^k - 1)$$

y

$$p = M + 2^{m-1}(2^k - 1)$$

Si se cumplen las condiciones de que todo factor primo de *M* divida a *N*, y que *p* sea primo con *N*, la complejidad lineal de la secuencia de salida $\{z(t)\}$ será $M \times n$. En la práctica deberemos elegir $M = N$. En este caso la condición de que *p* y *N* sean primos entre sí, se cumplirá si, y sólo si, *k* es primo relativo con *m*. Por tanto, si *m* es un número primo y $m = n$, entonces cualquier valor de *k* dará a la salida una secuencia de complejidad lineal $M \times n$ y periodo $M \times N$.

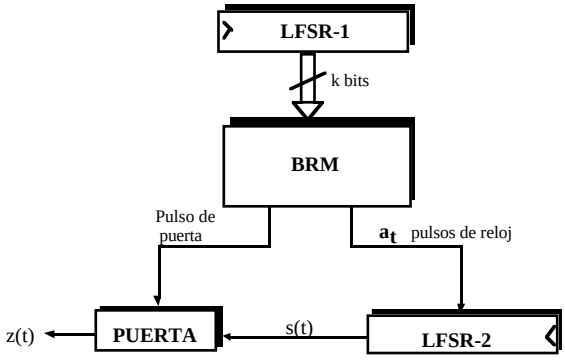

Fig. 4.14 Generador multiplicador de tasa binaria

Veamos un ejemplo. Supongamos que ambos registros de desplazamiento tienen 127 celdas y tienen polinomio de realimentación primitivo. Con estos valores, el periodo *M* será 2^{127}-1 y la complejidad lineal $127(2^{127}\text{-}1) = 2 \times 16 \times 10^{40}$, que es un valor considerablemente superior que el valor 127 que obtendríamos con un único registro de desplazamiento del mismo tipo. El periodo del generador BRM en este caso será $(2^{127}\text{-}1)^2 = 2{,}895 \times 10^{76}$.

4.2.2 Generador de secuencias autodecimadas

Los generadores de secuencias autodecimadas se basan en producir el avance del reloj que controla al generador en función del valor del bit de la propia secuencia de salida. A este proceso se le denomina decimación del registro de desplazamiento. Vamos a ver dos generadores que se basan en esta

técnica. El primero de ellos (generador de Ruepple) realiza una decimación de la secuencia de salida por un valor constante *d* que consiste en tomar muestras de la secuencia de salida a una velocidad *d* veces menor que el cambio de estado interno del registro de desplazamiento con realimentación lineal. El segundo generador (generador de Chambers-Gollmann) realiza una decimación (*d*,*k*) que, a diferencia de la anterior no es constante, sino que se realiza por una función dependiente de los dígitos previos de la secuencia del LFSR generador por lo que se obtiene como consecuencia una secuenia (*d*,*k*)-decimada.

Si bien en el capítulo anterior ya se vieron las propiedades de la decimación de secuencias, veamos algunas consideraciones antes de pasar a ver los generadores basados en esta técnica.

Convencionalmente, se define la decimación de una secuencia $s(t)$ $t = 0, 1, 2,...$ por un valor constante *d* como la extracción de cada *d*-ésimo dígito de $s(t)$, y usualmente se denota por $s[d]$. Por el contrario, también es posible realizar decimaciones no constantes sobre la secuencia, de forma que dependa de porciones previas de ésta. Las secuencias producidas de esta forma se denominan [*d*,*k*]-autodecimadas. Veámoslo con un ejemplo:

EJEMPLO 4.1: Sea la siguiente porción de una *m*-secuencia $s(t)$= 1111100100110000101101010 0..., producida por un registro de desplazamiento con realimentación lineal. Una secuencia 2-autodecimada a partir de la secuencia dada será:

$$s[2]= 1110010011000...$$

mientras que una secuencia [1,2]-autodecimada será:

$$s[1,2]=11101010000110110....$$

Para realizar el estudio de las secuencias autodecimadas se supondrá que las secuencias originales son *m*-secuencias (secuencias binarias de máximo periodo producidas por registros de desplazamiento con realimentación lineal). Esto implicará que $0 < d,k < 2^L-1$, si *L* es el número de celdas del LFSR. Sin embargo, para eliminar la restricción anterior bastará que aquellos *d* ó *k* que sean mayores que 2^L-1 se reduzcan mediante la operación mod 2^L-1. Las secuencias [*d*,*k*]-autodecimadas son tan fáciles de implementar como las propias *m*-secuencias, y si los valores de *d* y *k* se eligen adecuadamente, presentan tan buenas propiedades de distribución (excelente *k*-distribución) como éstas, con la ventaja adicional que tienen una complejidad lineal muy elevada, de valor similar al periodo, por lo que parecen comportarse de una forma más aleatoria que las *m*-secuencias. Estos generadores pueden tener buena aplicación en modulaciones *spread spectrum* y criptografía (ambas aplicaciones se estudiarán en el capítulo 7). Sin embargo, se debe tener precaución, ya que si para realizar un cifrado se usa una sola secuencia [*d*,*k*]-autodecimada a partir de una *m*-secuencia y el par [*d*,*k*] se hace público, un posible atacante podría obtener el polinomio de realimentación y el estado inicial a partir de la secuencia de salida mediante la resolución de un sistema de ecuaciones lineales.

A) Generador de secuencias autodecimadas de Ruepple

La estructura de este generador de secuencias pseudoaleatorias, propuesto por R.A. Ruepple en 1987 [RUE88] es muy simple, tal como muestra la figura 4.15. Se basa en un único registro de

desplazamiento con realimentación lineal con polinomio de realimentación primitivo, y cuya secuencia de salida se usa para controlar el propio reloj que marca la producción de los bits, de forma que si a la salida aparece un 0, el reloj produce *d* pulsos, y si aparece un 1, el reloj produce *k* pulsos. El efecto que causa esto es que la *m*-secuencia que produce el LFSR es decimada para formar la secuencia de salida de acuerdo con una regla pseudoaleatoria que viene determinada por la propia *m*-secuencia. Este generador no es muy bueno para aplicaciones criptográficas.

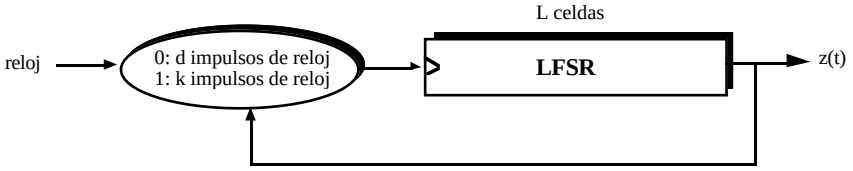

Fig. 4.15 Generador de secuencias autodecimadas de Ruepple

B) Generador de secuencias autodecimadas de Chambers y Gollmann

Otro generador basado en la autodecimación de secuencias es el propuesto por Chambers y Gollmann [CHA88] como una modificación del generador de secuencias autodecimadas de Ruepple, y cuyo esquema se muestra en la figura 4.16. El propósito final de este generador es disponer de un elemento básico a partir del cual construir estructuras mayores mediante cascada. Veamos cuál es su estructura y cuáles son sus propiedades: este generador está formado por un registro de desplazamiento con realimentación lineal con polinomio de realimentación primitivo cuyo reloj está controlado por el resultado de aplicar una función NOR sobre las *p* celdas de menor peso. Si el resultado de esta función es un 0, el registro de desplazamiento da un solo paso, y si es un 1, da dos pasos.

Con este generador se pretende conseguir que la secuencia de salida presente un periodo similar al máximo posible $T = 2^L-1$ para un registro de desplazamiento realimentado linealmente de L celdas y una complejidad lineal igual a T, o a $T-1$. Sin embargo, esto es muy delicado, puesto que a medida que aumenta p, las propiedades estadísticas de la secuencia autodecimada resultante se alejan más y más de las de la *m*-secuencia producida por el LFSR original. Hay, pues, que tener en cuenta ciertas restricciones entre el número de celdas (L) del LFSR, y el número de celdas que se emplean como entrada de la función NOR (p) para conseguir que la complejidad lineal de la secuencia de salida sea igual a su periodo, y ambos estén próximos al valor máximo obtenible 2^L-1. La restricción que debe cumplir el par (p, L) para obtener estas condiciones viene determinado por una relación respecto al periodo, expresada por:

$$T = (2^L - 1) - \frac{2(2^{L-p} - 1)}{3}$$

si (L-p) es par (condición necesaria para que el periodo T sea impar) y tanto el periodo T como la complejidad lineal son números 2-primos. Un número 2-primo es el que satisface que 2^i-1 no es

múltiplo de T para cualquier $i < T$-1. Una manera rápida de saber si T es 2-primo se basa en el hecho de probar los i, tales que valgan $(T\text{-}1)/T'$, si T' es cada uno de los los factores primos de T-1. La condición anterior se encuentra cuando se estudian cuántos ceros y unos se deben restar (en total $2\left(2^{L-p}-1\right)/3$) al periodo de la m-secuencia (2^L-1). Cuanto más parecidos son L y p, el periodo de salida del generador (T) y, por tanto, también su complejidad, tanto más se acercarán al máximo valor posible 2^L-1.

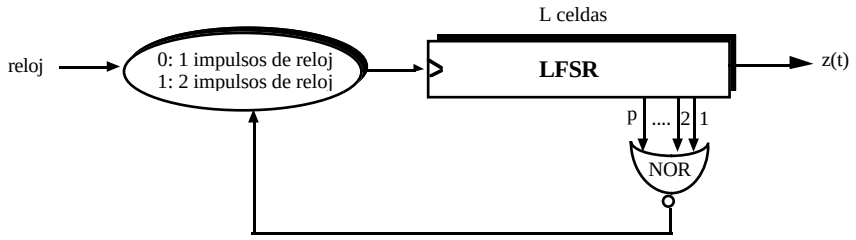

Fig. 4.16 Generador de Chambers-Gollmann

Para estudiar el comportamiento de este generador podemos hacer una comparación entre la secuencia producida por el generador (con la función NOR) y la m-secuencia original. Si para ello elegimos un LFSR de 5 celdas (L = 5) con polinomio de realimentación primitivo estas secuencias obtenidas son:

m-secuencia: 100<u>0</u>1101110101000<u>0</u>1<u>0</u>01<u>1</u>011001<u>1</u>111 (periodo = 31)

generador: 10011011101010000001100111 (periodo = 26)

Se puede ver que la secuencia obtenida por el generador se obtiene a partir de la m-secuencia pero eliminando los bits subrayados, ya que el generador se los ha saltado al dar dos pasos cuando ha encontrado la condición de p ceros consecutivos en la secuencia de salida.

Una cuestión importante a la hora de diseñar un generador de este tipo es que si el vector inicial coincide con uno de los vectores que se saltan pueden darse preámbulos en la secuencia antes de que se convierta en periódica. Estos preámbulos son siempre indeseables ya que reducen el periodo de la secuencia de salida en su misma longitud. Afortunadamente, se sabe que la máxima longitud que puede tener un preambulo es $\lfloor L - p + 1 \rfloor$ (donde $\lfloor x \rfloor$ indica la parte entera de x). Se puede estudiar cuántos estados iniciales hay globalmente buenos para evitar los preámbulos. Por ejemplo, para un LFSR con L = 5 y periodo T = 29, el número de polinomios primitivos de realimentación con periodo 31 es $\phi(2^L - 1)/L = 6$, y por lo tanto hay como mucho 6(31-29) = 12 estados que producirán preámbulos como mínimo en uno de esos polinomios de realimentación. Por tanto, se dispone de 31-12 = 19 estados iniciales globalmente buenos.

Para concluir el estudio de este generador, la tabla 4.2 muestra las combinaciones (L,p) para $L \leq 20$ que cumplen que el periodo de salida del generador (T) es un 2-primo. Se puede observar que,

aunque el límite teórico ofrece un rendimiento $T/2^L-1 = 66\%$, se garantizan rendimientos por encima del 80% ($4 < L \leq 70$), e incluso no es difícil encontrar rendimientos por encima del 99%. Por tanto, se puede decir que para valores $4 \leq L \leq 20$ se puede garantizar la obtención de secuencias con periodos y complejidades lineales del mismo valor y muy cercano el máximo teórico $T_{max} = 2^L-1$.

Tabla 4.2 Combinaciones (L,p) para $L \leq 20$

L	p	*T*	*$T/2^L-1$*
3	1	5	0,625
4	2	13	0,812
5	3	29	0,906
6	2	53	0,828
6	4	61	0,953
9	7	509	0,994
10	2	853	0,833
11	3	1877	0,916
12	2	3413	0,833
12	10	4093	0,999
14	12	16381	0,999
16	6	64853	0,989
17	5	128341	0,979
18	2	218453	0,833
18	10	261973	0,999
18	14	262133	0,999
19	7	521557	0,994
20	18	1048573	0,999

4.2.3 El generador de marcha y parada (*Stop & Go*)

Este tipo de generadores de secuencias pseudoaleatorias también se basa en controlar los impulsos de reloj que actúan sobre uno o varios generadores de *m*-secuencias, usando para ello una secuencia controladora de pasos. Veamos algunos generadores que se basan en esta técnica.

A) Generador de marcha y espera de Beth-Piper

Este generador de secuencias pseudoaleatorias [BET85] se basa en la aplicación de las técnicas de control de reloj descritas en el capítulo 3, y su estructura se puede ver en la figura 4.17. Su funcionamiento es el siguiente: El reloj que actúa sobre el registro de desplazamiento con realimentación lineal LFSR-2 está controlado por la salida del LFSR-1, de forma que el LFSR-2

cambiará de estado en el instante *t*, si y sólo si, la salida del LFSR-1 en el instante anterior es un 1 ($x_1(t\text{-}1) = 1$). Además, la salida del LFSR-2 se combina con la salida de un tercer LFSR (LFSR-3) y funciona de forma normal. Esta combinanción se realiza mediante una suma, bit a bit, de la salida de los LFSR 2 y 3, con lo que se obtiene la salida global del generador *z*(t). Si se imponen las restricciones adecuadas sobre los grados de los tres LFSR (n_1, n_2 y n_3 para los LFSR-1, LFSR-2 y LFSR-3 respectivamente) y sus polinomios son primitivos, $z(t)$ tendrá periodo P y complejidad Λ:

$$P = (2^{n_1} - 1)(2^{n_2} - 1)(2^{n_3} - 1) \qquad \Lambda(z) = (2^{n_1} - 1)n_2 + n_3$$

Sin embargo, aunque la complejidad lineal de la secuencia de salida del generador es elevadísima, este generador presenta ciertas dependencias estadísticas entre las secuencias que producen los diversos LFSR´s, que hace que su uso no sea adecuado en criptografía, ya que esta debilidad se puede emplear para criptoanalizar el sistema. Veamos esto: Sea $x_2`(t)$ la secuencia de salida del LFSR-2 en el instante *t* si suponemos a éste en libre funcionamiento (fuera del conjunto global y, por tanto, con pulsos de reloj normales). Entonces tendremos la siguiente probabilidad de coincidencia:

$$p = Prob[z(t) \oplus z(t+1) = x_3(t) \oplus x_3(t+1)] =$$
$$= Prob(x_1(t) = 0) + [Prob(x_1(t) = 1) \cdot Prob(x_2'(t) = x_2(t+1))] = \frac{1}{2} + \frac{1}{4}$$

El que esta probabilidad sea superior a 1/2 implica que si los polinomios de realimentación de los LFSR-1 y LFSR-3 son conocidos por un posible atacante, éste puede aplicar un ataque basado en el método del síndrome lineal [ZEN89], que permita primero recuperar la secuencia $x_3(t)$ a partir de la secuencia de salida $z(t)$, y después la secuencia $x_1(t)$ a partir de la $x_2(t)$. Ello se podría hacer incluso bajo la suposición de que el polinomio de realimentación del LFSR-2 no fuera conocido.

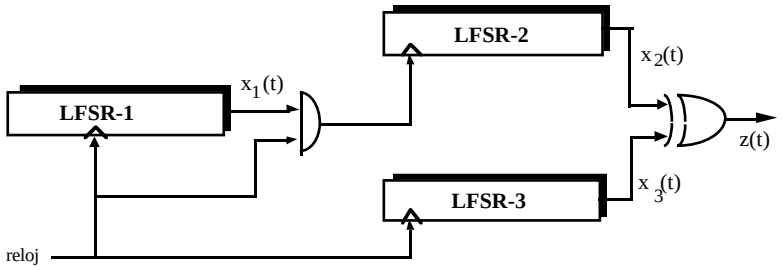

Fig. 4.17 Generador de Beth-Piper

B) Generador de marcha y espera en cascada de Gollmann

Este generador de secuencias pseudoaleatorias [GOL85] consiste en una variación del generador de Beth-Piper visto en el apartado anterior, y su esquema se puede ver en la figura 4.18.

Consiste en una serie de *N* registros de desplazamiento con realimentación lineal y polinomio de realimentación primitivo todos ellos del mismo grado *n*. El funcionamiento de este generador es el siguiente: El reloj que marca el funcionamiento del LFSR *i*-ésimo está controlado por todos los LFSR *j*-ésimos (con $j < i$ de forma conjunta). Este control se realiza de la misma manera que se vio en el generador de marcha y espera de Beth-Piper.

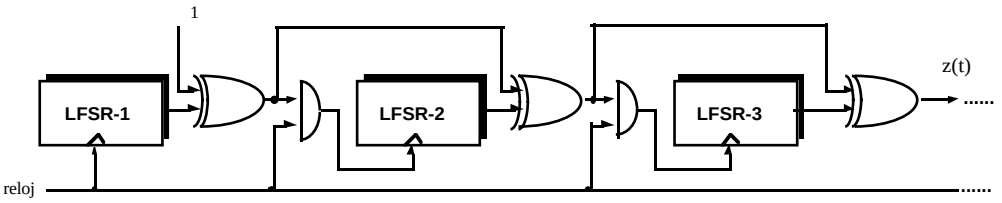

Fig. 4.18 Generador de marcha y espera en cascada de Golmann

Mediante esta técnica se consigue que la secuencia de salida $z(t)$ tenga el siguiente periodo:

$$T = (2^n - 1)^N$$

y una complejidad lineal muy elevada, acotada inferiormente mediante la expresión:

$$\Lambda(z) \geq n(2^n - 1)^{N-1}$$

C) Generador de marcha y espera de control bilateral

Este generador de marcha y espera [ZEN90] se basa en el hecho de que si una secuencia binaria *s* tiene un periodo *T* que es un primo impar, su complejidad lineal $\Lambda(s)$ estará acotada inferiormente por el orden del número 2 módulo *T* ($\mathrm{Ord}_T(2)$), es decir:

$$\Lambda(s) \geq Ord_T(2)$$

Cuando se use este generador en aplicaciones como la criptografía, el hecho de elegir una secuencia de periodo primo es deseable, debido a que se puede someter a diversas transformaciones sin que se modifique la cota inferior de su complejidad lineal, suponiendo que la transformación utilizada no nos conduzca a una secuencia constante. Sin embargo, si el periodo $T = 2^n\text{-}1$ resultara ser un primo de Mersenne, entonces $\mathrm{Ord}_T(2) = n$, y ello implica que el periodo *T* que elijamos debe ser un primo superior a $2^n\text{-}1$. De este modo podremos construir secuencias con periodos de la forma $T = q\ 2^n\text{-}1$, si *q* es un número impar mayor o igual que 3, y hay una estructura formada por dos registros de desplazamiento con realimentación lineal (LFSR) ambos de *n* celdas [ZEN90].

Un generador de estas características se muestra en la figura 4.19. Con esta estructura se puede obtener un periodo $T = 5 \cdot 2^{n-2} - 1$. Para ello, ambos LFSR de *n* celdas deben inicializarse con

estados iniciales distintos de cero. El control de los pasos en este generador se realiza de la siguiente forma:

(a) Si $(x(t+n\text{-}1),x(t+n\text{-}2)) = (0,1)$ se bloquea el pulso de reloj del LFSR-2.

(b) Si $(y(t+n\text{-}1),y(t+n\text{-}2)) = (0,1)$ pero $(x(t+n\text{-}1),x(t+n\text{-}2)) \neq (0,1)$, se bloquea el pulso de reloj del LFSR-1.

El diagrama de estados de este generador consta de $3 \cdot 2^{n-2} - 1$ ciclos ramificados de longitud $5 \cdot 2^{n-2} - 1$, cada uno de los cuales presenta un cierto número de ramas de longitud uno. Los valores posibles de n que hacen que $5 \cdot 2^{n-2} - 1$ sea primo se pueden determinar usando el criterio de Lucas-Lehmer o bien mediante un algoritmo que se puede encontrar en [ZEN90]. En este generador, la complejidad de la secuencia de salida es del orden de magnitud del periodo.

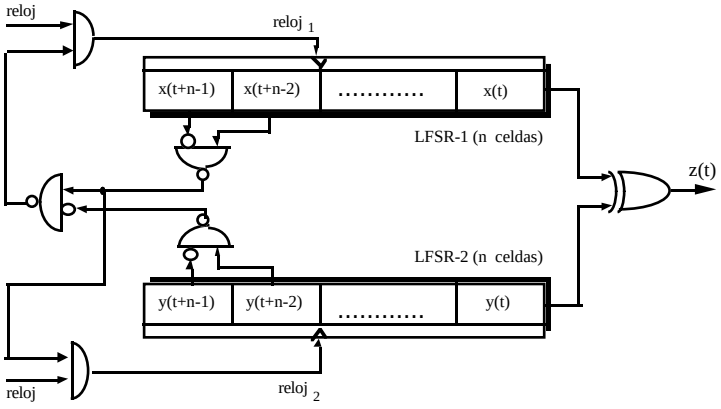

Fig. 4.19 Generador de marcha y espera de control bilateral

4.2.4 El generador de pasos alternados (ASG)

Este generador de secuencias pseudoaleatorias, propuesto por Günter en 1988 [GÜN88], comparte las ventajas de los generadores de marcha y espera vistos anteriormente, pero evita algunas de sus debilidades. Su estructura está formada por tres subgeneradores basados en registros de desplazamiento con realimentación lineal y su funcionamiento consiste en que la salida de uno de los LFSR (el LFSR-3 en la figura 4.20) controla el reloj que ataca a los otros dos registros de desplazamiento (LFSR-1 y LFSR-2). Este control se realiza de la siguiente forma: cuando el bit de salida del LFSR-3 es un 1, se le da un impulso de reloj al LFSR-1, y cuando es un 0, se da el pulso al LFSR-2. Si $x_1(t)$, $x_2(t)$ y $x_3(t)$ son las secuencias que producen los registros de desplazamiento LFSR-

1, LFSR-2 y LFSR-3 respectivamente, cuando se encuentran separados del generador (en funcionamiento independiente), la salida del generador ASG, $z(t)$, se obtiene sumando bit a bit módulo 2 las salidas de los LFSR-1 y el LFSR-2 ($z(t)= x_1(t) \oplus x_2(t)$).

En la práctica, los subgeneradores LFSR-1, LFSR-2 y LFSR-3 son generadores de *m*-secuencias (registros de desplazamiento con realimentación lineal con polinomio de realimentación primitivo). Sin embargo, en el artículo original de Günter el subgenerador 3 estaba constituído por un generador de secuencias de De Bruijn, debido a que ello simplificaba mucho su estudio teórico. Veamos algunos de los resultados obtenidos para este generador si el subgenerador 3 es un generador de secuencias de De Bruijn. Sin embargo, otras pruebas realizadas demuestran que los resultados usando un LFSR para el subgenerador 3 son muy similares.

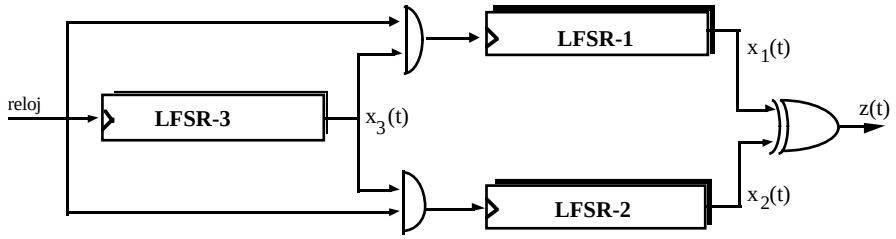

Fig. 4.20 Generador de pasos alternados ASG

Se estudiarán en primer lugar las características de periodo y complejidad lineal. Para facilitar la demostración se recurre a los siguientes teoremas:

TEOREMA 1: Si se efectúan las siguientes suposiciones:

(a) $x_3(t)$ es una secuencia de De Bruijn de periodo 2^k (si k es su número de celdas).
(b) Los polinomios de realimentación del LFSR-1 y del LFSR-2 son irreducibles y diferentes y tienen grados n_1 y n_2, y periodos T_1 y T_2 respectivamente.
(c) Los periodos T_1 y T_2 son mayores que 1 y primos entre sí: T_1, T_2 >1 y mcd(T_1,T_2) = 1.

Bajo estas suposiciones, el periodo de la secuencia de salida $z(t)$ del generador es:

$$T = 2^k \cdot T_1 \cdot T_2$$

y la complejidad lineal está acotada por la expresión:

$$(n_1 + n_2)2^{k-1} < \Lambda \leq (n_1 + n_2)2^k$$

TEOREMA 2**:** Si se considera la frecuencia de aparición de las subsecuencias cortas se pueden efectuar las siguientes suposiciones:

(a) $x_3(t)$ es una secuencia de de Bruijn de periodo 2^k (si k es su número de celdas).
(b) $x_1(t)$ y $x_2(t)$ son m-secuencias de periodos $T_1 = 2^{n_1} - 1$ y $T_2 = 2^{n_2} - 1$ respectivamente.
(c) Los periodos T_1 y T_2 son mútuamente primos entre sí (mcd(T_1,T_2) = 1.

Bajo estas suposiciones, la frecuencia de cualquier subsecuencia de longitud $w \leq min\{n_1, n_2\}$ es 2^{-w}, con un error del orden de $O\left(1/2^{n_1-w}\right) + O\left(1/2^{n_2-w}\right)$. La desviación de esta distribución respecto a la ideal es similar a la que presenta una m-secuencia. Este generador permite producir secuencias pseudoaleatorias de forma simple y eficiente, además de ser rápido.

Puede ser aconsejable formar una estructura en cascada utilizando este generador. Para ello, se sustituye cada LFSR por el generador de pasos alternados ASG. El generador así construído presentará elevado periodo, elevada complejidad lineal y buenas propiedades estadísticas en las secuencias obtenidas.

4.2.5 El generador multivelocidad de Massey-Ruepple

La estructura de este generador de secuencias pseudoaleatorias [MAS84] consiste en dos registros de desplazamiento con realimentación lineal (de grados m y n para el LFSR-1 y el LFSR-2 respectivamente, con $n \geq m$) con relojes funcionando a distinta velocidad, tal como muestra la figura 4.21. El reloj del LFSR-2 debe ser $d \geq 2$ veces más rápido que el del LFSR-1, con lo que la señal de salida del generador $z(t)$ vendrá dada por la expresión:

$$z(t) = \sum_{i=0}^{l-1} x(t+i)y(dt+i)$$

donde $x(t)$ e $y(t)$ son las secuencias producidas por el LFSR-1 y el LFSR-2 respectivamente.

Si los polinomios de realimentación utilizados por los registros de desplazamiento son primitivos, sus grados son mutuamente primos entre sí (mcd(m,n) = 1), y se cumple que el factor de velocidad d es mutuamente primo con el periodo del LFSR-2 (mcd(d,2^n-1) = 1), la secuencia de salida de este generador tendrá un periodo:

$$T = (2^n - 1)(2^m - 1)$$

y una complejidad lineal $\Lambda(z) = m \cdot n$. Además, la secuencia de salida producida por este generador presenta excelentes propiedades estadísticas.

Veamos cuál será la debilidad de este generador si se emplea en criptografía para realizar un sistema de cifrado en flujo. En este generador, el factor de velocidad d es variable y se puede emplear como parte de la clave secreta. La bilinearidad implicada en la generación de la secuencia de salida $z(t)$, fijado un valor del factor de velocidad, hace que el criptosistema se pueda romper por un ataque basado en el test de consistencia lineal [ZEN89]. Mediante este test, que se verá más detalladamente en el capítulo 7, si el atacante conoce los polinomios de realimentación de los registros de desplazamiento puede determinar el factor de velocidad d y los estados iniciales de los registros de desplazamiento (la

clave), aplicando tan sólo $(d_{max} - 1)(2^m - 1)$ tests de consistencia sobre una porción de la secuencia de salida de longitud $N \geq m + n + log_2 d_{max}$.

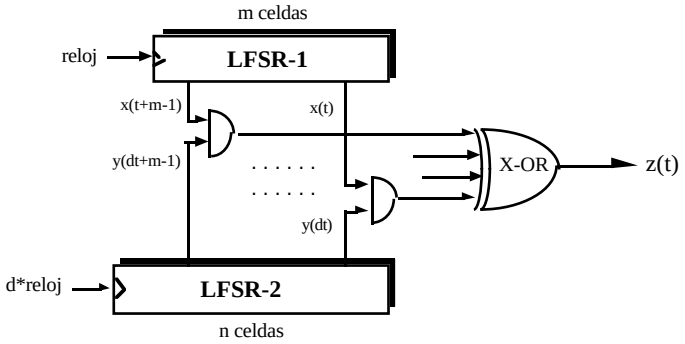

Fig. 4.21 Generador multivelocidad de Massey-Ruepple

4.2.6 El generador de producto interior

Consideremos dos registros de desplazamiento con realimentación lineal de L_1 y L_2 celdas respectivamente, cuyos relojes están atacados a velocidades diferentes (d_1 y d_2 respectivamente). La secuencia de salida $z(t)$ de este generador se obtendrá mediante el siguiente procedimiento:

FOR i = 1, 2,..... DO

Paso 1: Desplazar el $LFSR_1$ d_1 veces

Paso 2**:** Desplazar el $LFSR_2$ d_2 veces

Paso 3: Calcular el producto de los estados de los 2 generadores: $z_i = \sum_{i=1}^{min\{L_1, L_2\}} x_{1i} \cdot x_{2i}$

END FOR

donde las x_{ki}, con $k = 1, 2,..$ y $1 \leq i \leq min\{L_1, L_2\}$, son los contenidos de las celdas de ambos LFSR.

El periodo P de la secuencia de salida $z(t)$ estará acotado inferiormente por la expresión:

$$P(z) \geq P_1 P_2 / (q - 1)$$

y tendrá una complejidad lineal $\Lambda(z) = L_1 L_2$, suponiendo que los polinomios de realimentación de ambos registros de desplazamiento sean irreducibles, que los números de celdas sean mutuamente

primos entre sí (mcd(L_1,L_2) = 1), y además que mcd(d_1,P_1) = mcd(d_2,P_2) = 1, donde P_1 y P_2 son los periodos de los dos LFSR respectivamente.

El número de ceros en un periodo de la secuencia de salida será $(2^{L_1-1}-1)(2^{L_2}-1)$, y la diferencia entre el número de unos y ceros relativo a la longitud del periodo será $1/(2^{L_1}-1)$, considerando que los polinomios de realimentación sean primitivos y $L_1 > L_2$.

4.3 Generadores basados en otras técnicas

4.3.1 El generador de Windmill

El generador de Windmill [SMEET] presenta ventajas para la generación de secuencias pseudoaleatorias en modo bloque, es decir, para producir bloques de símbolos pseudoaleatorios. Además, presenta un paralelismo estructural que lo hace muy útil para realizaciones en VLSI (integración a muy alta escala). Un generador de Windmill consiste en una conexión en cascada cíclica de v ($v \leq 1$) registros de desplazamiento con realimentación lineal, tal como se muestra en la figura 4.22. Con cada LFSR se asocia un polinomio de realimentación $\alpha(t)$ y una red de realimentación lineal hacia el LFSR siguiente $\beta(t^{-1})$. A este conjunto se le llama un vano de Windmill. Según esta definición, el k-ésimo vano viene, pues, representado por el polinomio de realimentación hacia atrás $\alpha(t) = 1 - \sum_{j=1}^{m} \alpha_j t^j$ (ya que usa m celdas), y el polinomio de realimentación hacia adelante $\gamma_k(t) = t^{\lambda(k)}\beta(t^{-1})$ donde $\beta(t^{-1}) = \sum_{j=0}^{n} \beta_j t^{-j}$ (ya que usa n celdas), y $\lambda(k)$ representa el número total de celdas que tiene el registro de desplazamiento del vano, de forma que $\lambda(k) > \max(n,m)$. Cada vano tiene idénticas $\alpha(t)$ y $\beta(t^{-1})$. La primera celda del LFSR de cada vano se toma para formar un bloque de v bits y, para obtener la salida, éstos se deben combinar. La forma en que los v símbolos se combinan para producir la v-tupla final está controlada por una permutación σ. Así pues, la secuencia z de salida es:

$$\mathbf{z} = x_0^{\sigma^{-1}(0)},\ldots,x_0^{\sigma^{-1}(v-1)},x_1^{\sigma^{-1}(0)},\ldots,x_1^{\sigma^{-1}(v-1)},x_n^{\sigma^{-1}(0)},\ldots,x_n^{\sigma^{-1}(v-1)}$$

El generador completo se puede representar por $[\alpha(t),\beta(t^{-1}),\underline{\lambda},v,\sigma]$ donde $\underline{\lambda} = (\lambda(0),\ldots,\lambda(v-1))$ representa el vector que contiene los números de celdas de los v LFSRs del generador. Para cada vano k (k = 0, 1, ..., v-1), y para un $i \in N$ se tendrá un estado inicial $x_0^k, x_{-1}^k,\ldots,x_{-\lambda(k)+1}^k$ y la ecuación de recurrencia:

$$x_{i+1}^k = \sum_{j=1}^{m} \alpha_j x_{i+1-j}^k + \sum_{j=0}^{n} \beta_j x_{i+j-\lambda(k-1)+1}^{k-1}$$

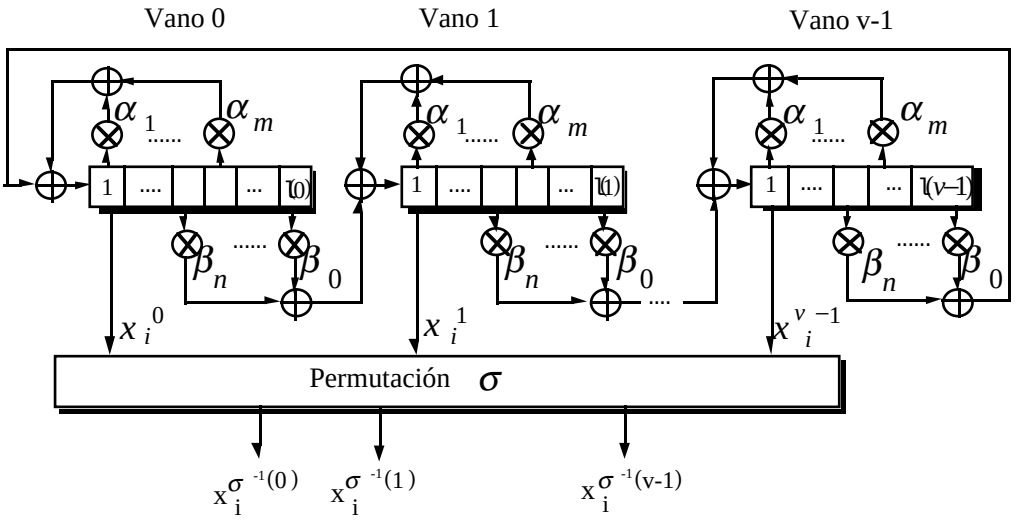

Fig. 4.22 Generador de Windmill

Podemos definir un polinomio de Windmill como $f(t) = \alpha(t^{\nu}) - \beta(t^{-\nu})t^{L}$, donde L y ν sean enteros tal que $1 \leq \nu < L$ y además sean mutuamente primos entre sí. Además, sean $\alpha(t)$ y $\beta(t^{-1})$ dos polinomios sobre GF(q) de grado positivo $m < L/\nu$ y $n < L/\nu$ respectivamente, de forma que $\alpha(0) = 1$ y $\beta(0) \neq 0$.

Con estas condiciones, si f(t) es un polinomio de realimentación primitivo sobre GF(q), existirá una permutación σ de los números 0, 1,..., ν-1, y un conjunto de longitudes de los registros de desplazamiento l definidas como:

$$\sigma(k) = Lk + c \pmod{\nu}$$
$$l(k) = (\sigma(k) - \sigma(k+1) + L)/\nu$$

donde k = 0, 1,..., ν-1, y c es un número fijo, la estructura $[\alpha(t), \beta(t^{-1}), l, \nu, \sigma]$ generará una m-secuencia z de la siguiente forma:

$$z(t) = \frac{\sum_{k=0}^{\nu-1} t^{\sigma(k)} p_k(t^{\nu})}{f(t)}$$

donde p_k vendrá definida como:

$$p_k = p_k(t) = x_0^k + \sum_{j=1}^{m}\sum_{i=1}^{j-1} \alpha_j x_{i-j}^k t^i + \sum_{j=0}^{n}\sum_{i=-l(k-1)+1}^{-j-1} \beta_j x_{i+j}^{k-1} t^{i+l(k-1)}$$

Por tanto, si el polinomio f(t) es primitivo, la secuencia $z(t)$ producida será una m-secuencia. Además, si el grado de $\beta(t^{-1})$ es igual a $\lfloor L/v \rfloor$, entonces al menos uno de los vanos tendrá su entrada conectada a la salida del vano mediante la realimentación hacia adelante. Esta conexión debe ser considerada cuidadosamente, ya que puede dar problemas de temporización en aplicaciones prácticas. A los polinomios de Windmill que no producen esta clase de conexiones, y evitan así problemas de temporización, se les llama polinomios propios de Windmill, y cumplen la restricción adicional $v(grad\beta(t^{-1}) + 1) \leq L$. Se puede hacer además, y sin pérdida de generalidad, que el parámetro c valga 0, de forma que los valores de $\gamma(k)$ y $\sigma(k)$ dependan sólo de L y de v. Veamos ahora el número de polimomios de Windmill f(t), tanto irreducibles (N_i) como primitivos (N_p), de que podemos disponer. Para ello, los consideramos como un subconjunto de los polinomios de grado L con f(0)=1. Bajo esta condición, el número de polinomios de Windmill irreducibles que satisfacen que f(0)=1 será de:

$$N_i = \frac{2^{1+2\lfloor L/v \rfloor}}{L}$$

y el número de polinomios de Windmill primitivos:

$$N_p = \frac{2^{1+2\lfloor L/v \rfloor}}{L} F(L)$$

donde $F(L) = \phi(2^L - 1)/2^L$ si ϕ es la función de Euler.

Podemos concluir sobre este generador que es bastante eficiente para generar secuencias binarias a elevadas velocidades y también para generar secuencias aleatorias en forma de bloque, por lo que sus aplicaciones pueden ser múltiples.

Este generador está propuesto por la *European Broadcasting Union* (EBU) para cifrar imágenes de televisión, usándolo de forma que genere bloques de símbolos pseudoaleatorios.

4.3.2 Generador de permutación de subsecuencias

Las secuencias producidas por este generador [JANSE] se obtienen mediante la permutación de subsecuencias tanto de unos como de ceros de una determinada secuencia binaria periódica $\underline{s} = (s_0, s_1, \ldots, s_{p-1})$ binaria ($s_i \in GF(2)$) de De Bruijn. Las secuencias de De Bruijn de orden n tienen un periodo de 2^n, tal como se vio en el capítulo 3. Esto implica que cada n-tupla ocurre exactamente una vez en la secuencia. De Bruijn [BRU46] demuestra que de dichas secuencias hay exactamente $2^{2^{n-1}-n}$, todas ellas con complejidad de máximo orden [JAN89].

Veamos a continuación cómo obtener familias de secuencias permutando las subsecuencias de unos y ceros de una secuencia de De Bruijn de un orden determinado. El procedimiento consiste en permutar de forma independiente los enteros que representan las subsecuencias de unos y los enteros que representan las subsecuencias de ceros. Una vez hecho esto, se vuelve a transformar la secuencia de enteros obtenida en una secuencia binaria.

El procedimiento descrito preserva las buenas propiedades respecto al cumplimiento del primer y segundo postulado de Golomb de la secuencia original y, además, la secuencia obtenida muestra un comportamiento excelente respecto a la complejidad de máximo orden [JAN89].

Si llamamos C_n a la clase de secuencias obtenidas de esta forma a partir de una secuencia de De Bruijn de orden *n*, el número de secuencias que tendrá esta clase vendrá determinado por la expresión:

$$|C_n| = 2^{-n+2} \prod_{l=1}^{n-2} \left(\frac{2^l}{2^{l-1}} \right)^2$$

Si los coeficientes binomiales de esta expresión se aproximan usando la fórmula de aproximación de Stirling, la ecuación queda como:

$$|C_n| = \rho_n \left(\frac{4}{\pi} \right)^{n-2} \prod_{k=1}^{n} G_k$$

donde $G_k = 2^{2^{k-1}-k}$ es el número de secuencias binarias de De Bruijn de orden *k* y ρ_n es un factor de corrección, menor que 1, independientemente del valor de *n*, y que converge hacia el valor 0,61... para valores de *n* grandes.

De la ecuación anterior se puede deducir que el número de secuencias de De Bruijn contenidas en la clase C_n tenderá a cero a medida que crezca *n*. Sin embargo, todas las secuencias de De Bruijn de orden *n* estarán contenidas en C_n, ya que por definición, están todas aquellas en las que las subsecuencias de longitud *n* ocurren exactamente una vez.

En lo que a la complejidad de máximo orden se refiere, puesto que las secuencias de De Bruijn de orden *n* (y, por tanto, las secuencias con complejidad de máximo orden *n*) son tan sólo una pequeña fracción de C_n, el resto de secuencias de la clase deberán tener necesariamente complejidades de máximo orden mayores que *n*. Se puede afirmar que para todas las secuencias $\underline{s} \in C_n$, $n > 2$, la complejidad de máximo orden $c(\underline{s})$ satisfará la inecuación $n \leq c(\underline{s}) \leq 2^{n-1} - 1$. Además, para $n > 2$ ninguna de ellas puede tener una $c(\underline{s}) = 2^{n-1} - 2$, y para $n \geq 4$ el número de secuencias del conjunto C_n que tendrán complejidad de máximo orden $c(\underline{s}) = 2^{n-1} - 1$ será $2^{-n+5}|C_{n-1}|$.

Veamos cómo generar este tipo de secuencias. Existe una forma eficiente de generarlas mediante el uso de técnicas de codificación enumerativas para generar las permutaciones [COV73]. Sin embargo, a pesar de su sencillez, la implementación de este método suele ser bastante compleja, por lo que a continuación se va a describir otro método más simple basado en LFSRs y contadores binarios que permite generarlas de una forma más sencilla. Este método parte del hecho de que no es necesario generar el conjunto C_n entero, sino tan sólo un subconjunto de éste, con secuencias que puedan aparecer más de una vez (por ejemplo, versiones desplazadas de una secuencia que ya apareció).

La figura 4.23 muestra un ejemplo de generador de permutación de subsecuencias de orden 5, capaz de generar 512 permutaciones diferentes de periodo 32. En dicha figura se pueden identificar dos generadores de De Bruijn de orden 3 basados en registros de desplazamiento realimentados no linealmente, uno para las subsecuencias de unos, y otro para las subsecuencias de ceros. Los

contenidos (estados) de las celdas de estos generadores de De Bruijn se permutan según cuál sea el contenido de los registros de clave.

El vector inicial permite que las dos secuencias de estado producidas se puedan combinar de 8 formas posibles. Las secuencias de estado producidas se pueden entonces convertir en secuencias de enteros mediante la utilización de la caja *S*. Esta caja *S* también se podría implementar de forma separada para ambas secuencias de estados, y permitir entonces la realización de permutaciones adicionales.

Tabla 4.4

Entrada	0	1	2	3	4	5	6	7
Salida	7	7	7	7	6	6	5	3

En la figura anterior, f representa la función no lineal de los generadores de De Bruijn donde $f(x_1,x_2,x_3) = x_1 + x_3 + \overline{x_2 \vee x_3}$, T es una báscula biestable que produce una secuencia de conmutación (10101010...) con una cadencia $c \wedge reloj$, c es la salida de un contador binario de 8 estados, y la caja *S* es la encargada de realizar una permutación de los bits según una tabla de sustitución (tabla 4.4).

Además, los generadores de De Bruijn tienen un reloj de entrada de velocidad $c \wedge reloj \wedge \underline{s}$, donde $\underline{s}$ es la secuencia de salida del generador.

Tabla 4.5

Estado gener. DB	*Entero*	*Gizq*	*Gder*
0 0 0	0	1	2
0 0 1	1	0	3
0 1 1	3	2	0
1 1 1	7	6	7
1 1 0	6	7	1
1 0 1	5	4	5
0 1 0	2	3	4
1 0 0	4	5	6

Veamos con un ejemplo cómo opera este generador. Para ello, se muestra una tabla (tabla 4.5) que contiene los estados sucesivos de los generadores de De Bruijn y los correspondientes valores enteros que servirán como entrada a la caja *S* teniendo en cuenta tanto la clave como el vector inicial. Veamos el proceso por el que se obtiene la secuencia de salida de este generador.

G derecho:	10 . 2 . 6 . . . . 7 . . . . . 4 . . . 3 . . 5 1
G izquierdo:	2 . 3 . 0 . . . 7 . . . . 1 . . 5 . . 4 . . . 6 . .
c_t:	7 . . . . 567 34 5 67 3456 7 . 67 6 7 . 6 767 5 6 7 . .
{s}:	0 1 0 1 0 1 1 1 0 0 0 0 0 1 1 1 1 1 0 1 1 0 0 1 0 0 1 1 0 0

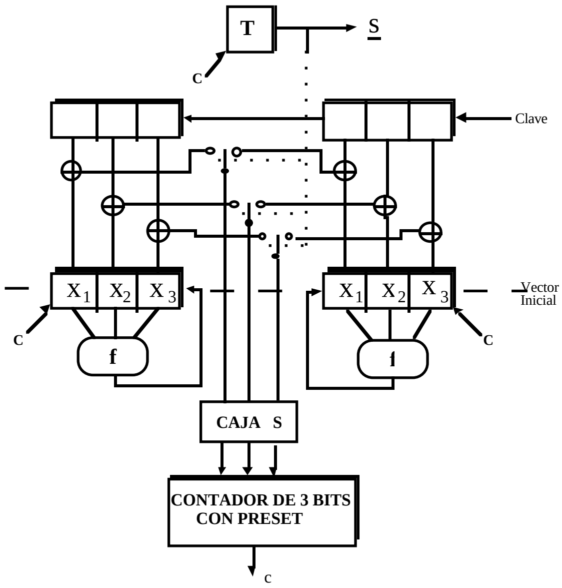

Fig. 4.23 Generador de permutación de subsecuencias

4.3.3 Generador basado en autómatas celulares

Los autómatas celulares han constituído un campo de gran interés desde los primeros estudios realizados por Von Neumann en los años 40. En 1985, Wolfram [WOL86][WOLFR] propuso el uso de autómatas celulares unidimensionales binarios con función de siguiente estado no lineal para generar secuencias pseudoaleatorias. Un autómata celular consiste en N celulas dispuestas en línea recta tal como muestra la figura 4.24. Denotemos los valores de las células en el instante t como a_1, a_2, ..., a_N, con un valor para cada célula de 1 ó 0. Estos valores se actualizarán en paralelo (sícronamente) en instantes de tiempo bien determinados de acuerdo con una regla fija que tendrá la forma

$$a'(k) = \Phi(a_{k-r}, a_{k-r+1}, \ldots, a_{k+r})$$

donde r denota el alcance de los argumentos de entrada a Φ. Cualquier implementación práctica de un autómata celular debe contener un número finito de celulas N. Estas presentan las disposiciones típicas en un registro circular con condiciones periódicas en los límites, es decir, los índices de los

argumentos de la regla del siguiente estado Φ se calculan módulo N. Una regla de actualización del siguiente estado interesante es la que viene determinada por $a'(k) = a_{k-1} \oplus (a_k \vee a_{k+1})$.

Veamos cómo funciona el generador de Wolfram. Para ello supongamos que tenemos N células (y posiblemente una función del siguiente estado Φ) cuyo estado inicial es $a(0) = (a_1(0), a_2(0), \ldots, a_N(0))$. El procedimiento es el siguiente:

FOR i = 1, 2,..... DO

Paso 1: Actualizar el autómata celular usando la siguiente regla:

$$a_k(i) = a_{k-1}(i-1) \oplus (a_k(i-1) \vee a_{k+1}(i-1))$$

para valores de k entre 0 y N - 1 y calcular módulo N los índices de las células.

Paso 2: Extraer la secuencia pseudoaleatoria $z(t)$ de cualquier único punto k:

$$z_i = a_k(i) \qquad i = 1, 2, \ldots\ldots$$

END FOR

El diagrama de transiciones entre estados de un autómata celular de N células consiste en un conjunto de ciclos, y se expresa mediante árboles que representan transiciones. Para este generador, el número de estados que tienen dos o más ceros predecesores tiende a cero asintóticamente con N. Para N grandes, el diagrama de transiciones entre estados aparece cada vez más dominado por un único ciclo. La longitud del ciclo máximo Π_N se puede aproximar por $\log_2 \Pi_N \approx 0.61(N + 1)$ para $N < 55$.

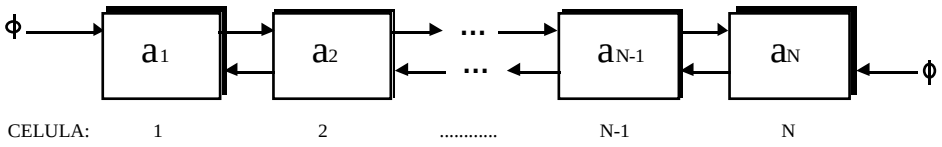

Fig. 4.24 Generador basado en un autómata celular

Además, al someter las secuencias producidas por este generador a una batería de tests estadísticos se observa que no presentan ninguna desviación significativa respecto a las secuencias verdaderamente aleatorias.

4.3.4 Generador de secuencias pseudoaleatorias basado en una memoria

Este generador, propuesto por J.D. Golic [GOLIC] [GOL90], es capaz de producir secuencias pseudoaleatorias mediente una función no lineal variante con el tiempo, aplicada sobre las celdas de

registros de desplazamiento con realimentación lineal (LFSR). Su estructura puede verse en la figura 4.25, donde se observa que está constituído por 3 LFSRs y una memoria de 2^K posiciones de 1 bit. Los tres LFSR deben tener longitudes m_i (i=1,2,3) y polinomios de realimentación primitivos. Los pulsos de reloj se aplican a los tres al mismo tiempo y, si su estado inicial no es el estado todo ceros, generarán m-secuencias de periodos $P_i = 2^{m_i} - 1$ (i=1,2,3). El contenido inicial de la memoria puede ser cualquiera. Las direcciones de *lectura* y *escritura* se tomarán de k celdas del $LFSR_2$ y k celdas del $LFSR_3$ respectivamente, mientras que la salida del $LFSR_1$ se usará para ir llenando la memoria. En cada instante t (t = 0,1,2,...) se producirán las siguientes operaciones:

Paso 1	El bit de salida del generador, $b(t)$, es leído de la posición de la memoria determinada por los k bits obtenidos de las celdas del $LFSR_2$ $x(t)$.
Paso 2	El bit de salida del $LFSR_1$, a(t), se escribirá en la posición de la memoria determinado por los k bits obtenidos de las celdas del $LFSR_3$ $y(t)$.

Debido al estado inicial de la memoria, las secuencias producidas por este generador no tienen por qué ser necesariamente periódicas, ya que presentarán preámbulos. Para hacer que sean periódicas e independientes del contenido inicial de la memoria, asumimos como instante de tiempo inicial $t_0 = P_3$ (de hecho, esto implica no considerar los preámbulos que se producirán antes de que la secuencia entre en una rutina periódica).

Para establecer unas cotas a la periodicidad y a la complejidad de las secuencias producidas por este generador, debemos efectuar las siguientes restricciones:$1 \le k \le \min(m_2,m_3)$ y $2^{m_3} - 1 \le m_1$. La salida del generador $b(t)$ se podrá expresar de la siguiente forma [GOL90]:

$$b(t) = \sum_{s=0}^{P_3-1} C_s(t) V_s(t) \qquad t = 0,1,2,...$$

donde

$$C_s(t) = \begin{cases} 1 & para \quad t - s = 0 \bmod P_3 \\ 0 & para \quad t - s \neq 0 \bmod P_3 \end{cases}$$

con s = 0, 1,..., P_3-1 y $V_s(t) = a(t - \phi_s(x_t))$ con t = 0, 1, 2,... y s = 0, 1,..., P_3-1.

x_t (t = 0,1,2...) es la secuencia de lectura, con periodo P_2, que puede tomar valores en el conjunto $K = \{0,1\}^k$. Para cada s = 0, 1,..., P_3-1, la función $\phi_s(j)$, ($j \in K$), es una aplicación inyectiva $K \rightarrow (1,...,P_3)$ [GOL90].

Si se cumplen las dos restricciones que se habían impuesto anteriormente, y se cumple además que:

a) $mcd(m_1, m_2) \neq m_1$

b) $mcd\left(P_2, \dfrac{P_1}{mcd(P_1, P_2)}\right) = 1$

c) $mcd(P_3, P_1 P_2) = 1$

este generador será capaz de producir $P_1P_2P_3$ secuencias distintas para todos los estados iniciales no nulos de los $LFSR_i$ (i = 1,2,3). El hecho de que este generador sea capaz de producir gran número de secuencias pseudoaleatorias distintas lo hace potencialmente útil para aplicaciones como la criptografía o las comunicaciones *spread spectrum*, que se tratarán más adelante.

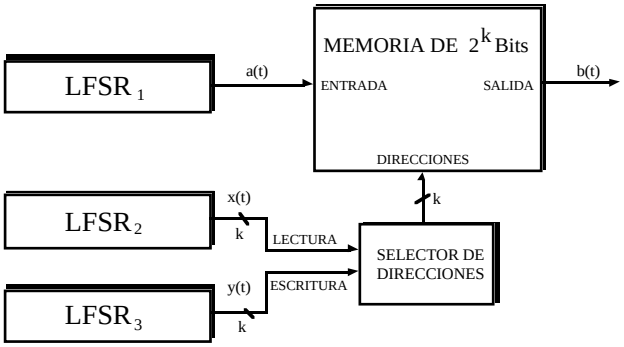

Fig. 4.25 Generador basado en una memoria

4.4. Problemas

PROBLEMA 4.1 Se desea obtener una secuencia pseudoaleatoria de periodo P = 259.967. Establecer los parámetros necesarios si se desea producirla mediante:

a) un generador de Jennings
b) un generador de umbral
c) un generador de Ruepple
d) un generador de marcha y espera de Beth-Piper
e) un generador multivelocidad de Massey-Ruepple
f) un generador de Pless como el de la figura 4.6

(Observar que 259967 = 127 × 2047).

PROBLEMA 4.2 Sea un generador de Geffe, como el de la figura 4.3, con 3 LFSRs de 3, 5 y 7 celdas respectivamente, y todos ellos con polinomio primitivo. Determinar el periodo y la complejidad lineal de la secuencia producida.

PROBLEMA 4.3 Obtener a partir de la secuencia binaria $s(t)$ = 11010111011011000101010011011:

a) una secuencia [2]-autodecimada
b) una secuencia [3]-autodecimada
c) una secuencia [1,2]-autodecimada
d) una secuencia [2,3]-autodecimada

PROBLEMA 4.4 En un generador de Ruepple, como el de la figura 4.9, con 3 LFSRs de 5, 7 y 11 celdas respectivamente y polinomios de realimentación primitivos, determinar:

a) el periodo de la secuencia de salida
b) su grado de inmunidad a la correlación
c) su complejidad lineal

PROBLEMA 4.5 Sea un generador ASG, como el de la figura 4.20, en el que el LFSR-3 se sustituye por un generador de De Bruijn de periodo 256, y los LFSR 1 y 2 tienen 13 y 21 celdas respectivamente y polinomio de realimentación primitivo. Para la secuencia de salida de esta estructura determinar:

a) el periodo
b) la frecuencia de aparición de una subsecuencia de 5 unos
c) la frecuencia de aparición de una subsecuencia de 1 uno
d) la frecuencia de aparición de una subsecuencia de 11 ceros
e) la frecuencia de aparición de una subsecuencia de 1 cero
f) acotar su complejidad lineal

4.5 Bibliografía

[AKLSG] AKL, S. G.; MEIJER, H. *A Fast Pseudo Random Permutation Generator With Applications to Cryptology*. Dept. of comput. and Inform Science. Queens University. Kingston. Ontario, Canada.

[ARA81] ARAZI, B. *On the Synthesis of De Bruijn Sequences*. Information and Control, Vol. 49, Nº 2, May 1981, pp.81-90.

[ARV77] ARVILLIAS, A. C.; MARITSAS, D. G. *Combinational Logicfree Realisations for High-Speed m-sequences Generation*. Electronic Letters, Vol. 13, nº 17, 1977, pp.500-502.

[BET85] BETH, T.; PIPER, F. *The Stop-and-Go Generator*. Advances in Criptology. Proc. Crypt'84. Springer-Verlag Lecture Notes in Computer Science, Nº 209, pp. 88-92, New-York, 1985.

[BLU82] BLUM, L.; BLUM, M.; SHUB, M. *Comparision of Two Pseudo Random Generators*. Proceedings of Crypto 1982.

[BLU84] BLUM, M. MICALLY, S. *How to Generate Cryptographically Strong Seqeunces of Pseudo Random Bits*. SIAM J. Computing 13, 4, pp. 850-864, Nov. 1984.

[BOY89] BOYAR, J. *Interferring Sequences Produced by Pseudo Random Number Generators*. J. ACM, Vol. 36, Nº. 1, Jan. 1989, pp. 129-141.

[BRU46] DE BRUIJN, N. G. A *Combinational Problem*. Nederl-Akad, Wetensch. Proc, Vol 49, pp 758-754, 1946.

[BRU82] BRUER, J.O. *On Nonlinear Combinations of Linear Shift Register Sequences*. Proc. IEEE Int. Symp, Jun. 1982, pp. 21-25.

[CHA84] CHAMB*ERS, W. G.; JENNINGS, S. M. Linear Equivalence of Certain BRM Shift-Register Sequences*. Electron. Lett., Vol. 20, pp. 1018-1019, Nov. 1984.

[CHA88] CHAMBERS, W. G. *Clock-controlled Shift Registers in Binary Sequences Generators.* IEEE Proc. Vol. 135, Pt. E, Nº 1, January 1988.

[CHA88] CHAMBER; GOLLMANN. *Generators for Sequences with Near-Maximal Linear Equivalence*. IEE Proc. Vol 135. Pt. E. Np 1, Jan 1988.

[CHA89] CHAMBERS; GOLLMANN. *Lock-in Effects in Cascades of Clock-Controlled Shift Registers*. Proc. Eurocrypt'88. Springer-Verlag Lecture Notes in Computer Science, Nº 330. New York 1989.

[CHA89] CHAMBERS; GOLLMANN. *Clock-Controlled Shift Registers: A Review*. IEEE J. on Selected Areas in Comm., Vol. 7, Nº 4, May 1989.

[COV73] COVER, T. *Enumerative Source Coding*. IEEE Trans. on Inform. Theory, Vol IT-19, pp. 73-76, January 1973.

[DAV74] DAVIO, M.; BIOUL, G. *Interconnection Structure of Cyclic Counters Made Up of JK Flip-flops*. M.B.L.E. Rep. R279, Brussels, Belgium, Dec. 1974.

[EBU76] *Specification of the Systems of the MAC/packet Family*. European Broadcasting Union. Tech 3258-E (Brussels:EBU Techinical Center), 1976.

[EUR88] EUROCRYPT. *Alternating Steps Generators by De Bruijn Sequences*. Proc. Eurocrypt'87. Springer-Verlag Lecture Notes in Computer Science, Nº 309. New York. 1988.

[FUS91] FUSTER; DE LA GUIA, NEGRILLO, MONTOYA. *Diseño e implementación de algoritmos de Generación de Secuencias Binarias*. I Reunión Española sobre Criptología. Mallorca, Oct. 1991.

[GEF73] GEFFE, P. R. *How to Protect Data with Ciphers that are Really Hard to Break*. Electronics, pp. 99-101, Jan. 4, 1973.

[GOL82] GOLOMB, S. W. *Shift Register Sequences*. Aegean Park Press, Laguna Hills, Calif., 1982.

[GOL84] GOLLMANN, D. *Linear Recursions of Cascaded Sequences.* Contributions to General Algebra, 3, Proc. of the Vienna Conference, June 1984. (Springer-Holder-Pichler-Tempsky, Wien 1985- Verlag B.G. Teuner, Stuttgart).

[GOL85] GOLLMANN, D. *Pseudo-Random Properties of Cascade Connections of Clock-controlled Shift Registers*. Advances in Cryptology. Proc. of Eurocrypt' 84, Vol. 209. Lecture Notes in Computer Science. Beth, Cot, and Ingemarsson Eds. Berlin. Springer-Verlag, 1985.

[GOL90] GOLIC, J. D.; MIHALJEVIC, M. M. *Minimal Binary Sequence Generator.* IEEE Trans. Inform. Theory. vol IT-36, pp. 190-192, Jan. 1990.

[GOL90] GOLIC, J.; MIHALJEVIC, M. M. *Minimal Linear Equivalent Analysis of a Variable-memory Binary Sequence Generator*. IEEE Trns. Inform. Theory, vol. IT-36, pp. 190-192, Jan. 1990

[GOLIC] GOLIC, J. D. *A Number of Output Sequences of a Binary Sequence Generator*.

[GRO71] GROTH. *Generation of Binary Sequences with controllable Complexity*. IEEE Trans. on Information Theory, Vol. 13, Nº 3, May 1971.

[GÜN88] GÜNTHER, C. G. *On the Properties of the Sum of Two Pseudo Random Sequences*. Proc. Eurocrypt'87, Lect. Notes in Comp. Science, Vol. 304, pp. 53-64, Springer-Verlag, 1988.

[GÜN88] GÜNTHER, C. G. *Alternating Step Generators Controlled by De Bruijn Sequences*. Proc. Eurocrypt'88. Springer-Verlag Lecture Notes in Computer Science, Vol. 304, pp. 5-14, New-York, 1988.

[HER82] HERLESTAM, T. *On using Prime Polynomial on Crypto Generators*. Proc. Workshop on Cryptography. Springer-Verlag Lecture Notes in Computer Science, Nº 149, New York, 1982.

[HOP79] HOPCROFT, J. E.; ULLMANN, J. D. *Introduction to Automata Theory*. Languajes and computation Reading, MA. Addison-Wesley, 1979.

[HUR74] HURD, W. J. *Efficient Generation of Statistically Good PseudoNoise by Linearity Interconnected Shift Registers*. IEEE Trans, 1974, C-23, pp.146-152.

[IEE80] IEEE. *Some Randomness Properties of Cascaded Sequences*. Trans. on Inform. Theory, Vol. IT-26, pp. 227-232, Marsch 1980.

[JAN89] JANSEN, C. J. A. *Investigations On Nonlinear Streamcipher systems: Construction and Evaluation Methods.* PhD. Thesis, Technical University of Delft, Delft, April 1989.

[JAN90] JANSEN, C. J. A. *Investigations on Non-Linear Streamciphers Systems: Construction and Evaluation Methods*. Ph. D. Thesis, Thecnical University of Delft, Delft, April 1989.[JANSE] JANSEN, C. J. A. *On the Construction of Run Permutated Sequences*. Philips Crypto B.V. Eindhoven, The Netherlands.

[JEN80] JENNINGS, S. M. *A Special Class of Binary Sequences*. Ph. D. Thesis. Westfield College, London University, 1980.

[JEN82] JENNINGS, S. M. *Multiplexed Sequences: Some Properties of the Minimum Polynolial.* Proc. Workshop on Cryptography, Nº 149. Springer-Verlag Lecture Notes in Computer Science, New York, 1982, pp.189-206.

[KEY76] KEY, E. L. An Analysis of the Structure & Complexity of Non-linear Binary Sequence Generators. IEEE Trans. on Inform. theory, Vol. IT-22, 1976.

[KNO76] KNOTT, G. D. *A Numbering System for Permutations and Combinations*. Communications of the ACM, Vol. 19, Nº 6, June 1976.

[KNU81] KNUTH, D. E. *The Art of Computer Programming. Volume 2: Seminumerical Algorithms.* Addison-Wesley, 1981.

[LEM71] LEMPEL, A.; EASTAMN, W. L. *High Speed Generation of Maximal Lenth Sequences*. IEEE Trans. on Comput., Vol. C-20, 1971, pp. 227-229.

[LID86] LIDL, R..; NEDERREITER, H. *Introduction to Finite Fields and Their Applications*. Cambridge University Press, 1986

[MAC65] MACLAREN, M. D.; MARSAGLIA, G. *Uniform Random Number Generators*. J. Ass. Comput. Machinery, 12, pp. 83-89, 1965.

[MACTE] *Mathematical Foundations of Flip-flops*. MAC Tech. Memo.47.

[MAR77] MARITSAS, D. G.; ARVILLIAS, A. C.; BOUNAS, *A. Phase-Shift Analysis of Linear Feedback Shift-Register Structures Generating Pseudo Random Sequences*. IEEE Transactions, 1977.

[MAS84] MASSEY, J. L.; RUEPPLE, R. A. *Linear Ciphers and Random Sequence Generators with Multiple Clocks*. Proc. Eurocrypt'84. Springer-Verlag Lecture Notes in Computer Science, New-York, 1984.

[MAY87] MAYER, A.; SKJERVEN, T. *Sequences from Self-Clocked Shift-Registers*. Semester Work, ETH-Zürich, 1987.

[MEI92] MEIER; STAFFELBATCH. *Correlation Properties of Combiners with Memory in Stream Ciphers*. J. of Cryptology, Vol. 5, Nº 1, 1992.

[ORE75] O´REILLY, J.J. *Series-parallel Generation of m-sequences*. Radio & Electron. Eng, 1975, 45, pp. 171-176.

[PLE77] PLESS, V. S. Encypting Schemes for Computer confidentiality. IEEE Trans. on Computer, Vol. C-26, Nº 11, 1977.

[PUI92] PUIG, M. *Estudio Comparativo de Cifradores en Flujo. Aplicaciones en Comunicaciones Móviles*. Proyecto final de Carrera. Universidad Politécnica de Cataluña (UPC), 1992.

[QUA74] QUAN, A. Y. C. *A Note on High Speed Generation of Maximal Lenth Sequences*. IEEE Trans., 1974, C-23, pp.201-203.

[RUB79] RUBIN. *Decrypting a Stream Cipher Based on JK flip-flops*. IEEE Trans. on Comput., Vol. C-28, Jul. 1979.

[RUE86] RUEPPLE, R. A. *Analysis and Design of Stream Cipher*. Springer-Verlag, NY, 1986

[RUE88] RUEPPLE, R. A. *When Shift Register Clock Themselves*. Proc. Eurocrypt'87. Springer-Verlag Lecture Notes in Computer Science, Nº 304, New York, 1988.

[SIE84] SIEGENTHALER, T. *Correlation-Inmunity of Non-linear Combinig Functions for Criptographic Applications*. IEEE Trans. on IT, Vol. IT-30, Nº 5, 1984.

[SHA83] SCHAMIR, A. *On the Generation og Cryptographically Strong Pseuod Random Sequences.* ACM Transactions on Computer Systems 1, 1, pp.38-44, Feb. 1983.

[SMEET] SMEETS, B. J. M.; CHAMBERS, W. G. Windmill Generators: *A Generalization and an Observation of How Many There Are*.

[STAFF] STAFFELBACH, O.; MEIER, W. *Analysis of Pseudo Random Sequences Generated by Cellular Automata*.

[SUR78] SURBÖCK, F.; WEINRITCHER, H. I*nterlacing Properties of Shift Register Sequences with Generator Polynomials Irreducible Over GF(p)*. IEEE Trans. on Inform. theory, Vol. IT-24, 1978, pp. 386-389.

[TAT89] TATEBAYASHI, M.; MATSUZAKI, D. B.; NEWMAN, JR. *A Cryptosystem Using Digital Signal Processors for Mobile Communications*. Proc. IEEE, 1989.

[TSA91] TSALIDES, PH.; YORK, T. A..; THANAILAKIS, A. *Pseudorandom Number Generators for CLSI Systems Based on Linear Cellular Automata*. IEE Proceedings-E, Vol. 138, Nº 4, July 1991.

[WOL86] WOLFRAM, S. *Random Sequence Generation by Cellular Automata*. Advances in Applied Mathematics 7, pp.123-169, 1986.

[WOLFR] WOLFRAM, S. *Cryptography with Cellular Automata*. Advances in Cryptology-Crypto'85, Proceedings, pp.429-432, Springer-Verlag, 1986.

[ZEN88] ZENG, K. C.; HUANG, M. Q. *On the Linear Syndrome Method in Cryptanalysis*. Proc. Crypto' 88, Springer-Verlag, Lecture Notes in Computer Science, Nº 403, New-York, 1988.

[ZEN89] ZENG, K. C.; YANG, C. H.; RAO, T. R. N. *On the Linear Consistency Test (LCT) in Cryptanalysis with Applications*. Proc. Crypto' 89, Springer-Verlag, Lecture Notes in Computer Science, Nº 403, New-York, pp. 469-478, 1989.

[ZEN90] ZENG, K. C.; YANG, C. H.; RAO, T. R. N. *Large Primes in Stream-Cipher Cryptography*. Proc. Auscrypt' 90, springer-Verlag Lecture Notes in Computer Science, Nº 453, New-York, pp.194-205, 1990.

[ZEN91] *ZENG; YANG; WEI; RAO. Pseudorandom Bit Generators in Stream-Cipher Criptography*. IEEE Computer. Feb. 1991.