Propuesta para la construcción de generadores de secuencias pseudoaleatorias basados en LFSR

Francisco I. Peralta, Gonzalo I. Duchén, Rubén Vázquez.

Instituto Politécnico Nacional
Escuela Superior Mecánica y Eléctrica Unidad Culhuacan
Sección de Estudios de Posgrado e Investigación
Av. Santa Ana, 1000, Col. San Francisco Culhuacan, C. P. 04430,
Coyoacan, México D. F.
franciscoipc@calmecac.esimecu.ipn.mx

Abstract. This paper shows the results of using the Berlekamp-Massey algorithm to construct an SLFSR (Shortest Linear Feedback Shift Register) that can help to reconstruct a plaintext that was encrypted using an LFSR. It is also used to produce polynomials using sequences of bits. Additionally, the design of a function that represents an LFSR (Linear Feedback Shift Register) and the implementation into an FPGA are made with a combination between hardware description language like VHDL, and the result that produce the algorithm of Berlekamp-Massey. This function, without being built from a primitive polinomial, permits the hardware implementation of stream ciphers based on LFSRs.

Keywords: LFSR, VHDL, Linear Complexity, Berlekamp-Massey Algorithm, Massey-Rueppel Generator, Primitive Polynomials.

Resumen: Este artículo presenta los resultados de utilizar el algoritmo de Berlekamp-Massey para romper un cifrado hecho a partir de un LFSR, así como también para crear polinomios a partir de una secuencia de bits. Además con la ayuda de un lenguaje de descripción de hardware combinado con el resultado que produce el algoritmo de Berlekamp-Massey, se puede crear en un FPGA la función de un Registro de Desplazamiento con Retroalimentación Lineal (LFSR: Lineal Feedback Shift Register). Esta función, sin ser construida a partir de un polinomio primitivo, permite crear en hardware arreglos de LFSR, que generan secuencias pseudoaleatorias con mejores propiedades criptográficas que las secuencias generadas por la función original.

Palabras Clave: LFSR, VHDL, Complejidad Lineal, Algoritmo de Berlekamp-Massey, Generador Multivelocidad, Polinomio Primitivo.

Introducción

En la actualidad la protección de información por medio de técnicas criptográficas es una práctica muy importante para la transferencia segura de datos. La criptografía se puede aplicar por ejemplo, para dar confidencialidad a un mensaje o garantizar la autenticidad de criptogramas y/o remitentes y destinatarios. Actualmente existen distintos tipos de cifrado que ayudan a otorgar atributos de seguridad a una comunicación, entre ellos el cifrado de flujo que cifra bit a bit los datos de un texto claro, utilizando una secuencia cifrante y una función booleana. Este cifrado, realizado con LFSR posee las siguientes propiedades importantes:

. Los LFSR se pueden realizar fácilmente en hardware.
2. Pueden generar secuencias de periodo T grande, con $T = 2^L-1$ y L la longitud del LFSR.
3. Producen secuencias con buenas propiedades estadísticas, que pueden evaluarse con los postulados de Golomb [7]:
4. Debido a su estructura, pueden analizarse usando técnicas algebraicas, una de estas la complejidad lineal.

En telecomunicaciones es muy utilizado con las siguientes consideraciones:
Cuando el espacio de almacenamiento es limitado.
Cuando los caracteres deben procesarse individualmente y recibirse de igual forma.
Cuando la propagación de errores está limitada.
Cuando el error de transmisión tiene una probabilidad alta.

2 LFSR (Registro de Desplazamiento con Retroalimentación Lineal)

Un LFSR de longitud L consiste en L etapas (o unidades de almacenamiento) numerados desde 0 hasta L-1, cada una capaz de almacenar un bit; tiene una entrada, una salida y un reloj que controla el flujo de datos entre las distintas etapas.
Durante cada unidad de tiempo se realizan las siguientes operaciones [1]:
1. El contenido de la etapa 0 es la que otorga el flujo de salida.
2. El contenido de la etapa i se traslada a la etapa i-1 considerando que, $1 \le i \le L-1$; y
3. El nuevo contenido de la etapa L-1 es el bit de retroalimentación s_j, que se calcula por medio de una función booleana que depende de la combinación de los contenidos de ciertas etapas predefinidas.

La figura 1 muestra la estructura general de un LFSR de longitud L. Cada c_i es 0 ó 1; el bit s_j es una combinación lógica no lineal del contenido de las etapas L, según sea el valor del vector {c}. Un LFSR puede generar secuencias pseudo-aleatorias de tamaño máximo 2^L-1. La secuencia resultante se denomina m-secuencia. Para que un LFSR tenga el periodo máximo (2^L-1), el polinomio formado por {c} más 1 debe ser un polinomio primitivo módulo 2 [3]. A continuación se dan algunas definiciones para conocer las características que deben cumplir los polinomios primitivos.

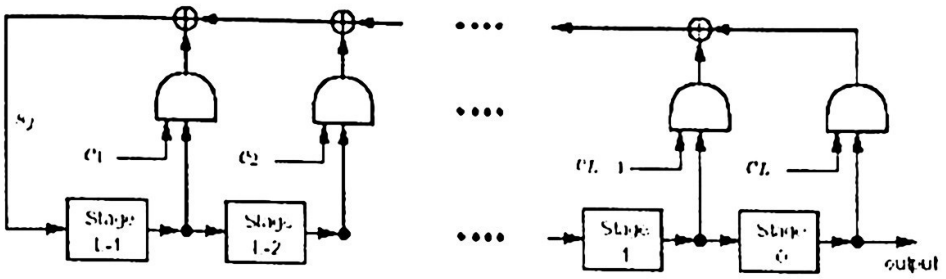

Fig. 1. LFSR de longitud L.

Definición. Si hay un polinomio p(x) tal que, r(x)a(x)=s(x), se dice que el polinomio s(x) es divisible por el polinomio r(x) o que r(x) divide a s(x). Un polinomio diferente de cero p(x) que solo es divisible por p(x) se dice irreducible ya que no puede ser escrito como el producto de dos polinomios, cada uno de grado positivo [6].

Definición. Un polinomio irreducible p(x) de grado L se dice que es primitivo solo si
1) p(x) divide a $X^n + 1$, con $n = 2^L - 1$, y
2) p(x) no divide a $X^n + 1$, cuando $n < 2^L - 1$.

3 Complejidad Lineal

Que un LFSR genere una secuencia con periodo máximo no implica que el cifrar alguna información con ella sea seguro. La complejidad lineal es una métrica importante para analizar a los LFSR y los generadores basados en LFSR, ya que mide la robustez o seguridad de un generador de secuencias pseudoaleatorias o cifrador de flujo. Concretamente, la complejidad lineal se define como la longitud L, del LFSR más pequeño o SLFSR (Shortest Linear Feedback Shift Register) que puede imitar la salida del generador. A continuación se darán las definiciones formales relacionadas con la complejidad lineal.

Definición. Sea $s = s_0, s_1, s_2, \ldots$ una secuencia infinita. La subsecuencia que consiste en los primeros n términos de s se denota por $s^n = s_0, s_1, s_2, \ldots, s_{n-1}$.

Definición. Un LFSR genera una secuencia s si hay un estado inicial para el cual secuencia de salida del LFSR es s. De forma similar, un LFSR genera una secuencia finita s^n si hay algún estado inicial para el cual la secuencia de salida del LFSR tiene s^n como sus primeros n términos.

Definición. La complejidad lineal L(s), de una secuencia binaria infinita s, se define como:
1) Si s es la secuencia cero s = 0, 0, 0,..., entonces L(s)=0;
2) Si ningún LFSR genera s, entonces $L(s)=\infty$;
3) De otra forma, L(s) es la longitud del LFSR más pequeño que genera s.

3.1 Propiedades de la Complejidad Lineal

Sean s y t secuencias binarias.
1) Para cualquier $n \geq 1$, la complejidad lineal de la subsecuencia s^n satisface $0 \leq L(s^n) \leq n$.
2) $L(s^n) = 0$ si y solo si s^n es la secuencia cero de longitud n.
3) $L(s^n) = n$ si y solo si $s^n = 0, 0, 0, ..., 0, 1$.
4) Si s es periódica con periodo N, entonces $L(s^n) \leq N$.
5) $L(s \oplus t) \leq L(s) + L(t)$, donde $s \oplus t$ denota la operación XOR de los bits de s y t [1].
Se sabe que el SLFSR generado a partir de una secuencia tiene una gran importancia práctica, en especial en criptografía y teoría de códigos. El algoritmo de Berlekamp-Massey es eficiente para obtener el SLFSR de una secuencia y se desarrolla a continuación.

4 Algoritmo de Berlekamp-Massey

El algoritmo de Berlekamp-Massey (ABM) hace uso de la complejidad lineal de una secuencia binaria finita s^n de longitud n para medir la robustez de un cifrador de flujo [4]. El algoritmo toma k iteraciones, con la k-ésima iteración escribe la complejidad lineal de la subsecuencia s^k que consiste en los primeros k términos de s^n [1]. Además, puede generar el SLFSR después de examinar solo 2L bits de la secuencia de salida. Donde L es la longitud del LFSR. Una vez generado, se podrá romper el cifrador ya que se conoce el polinomio con el cual se construye el SLFSR [3].

Definición. Considere la secuencia binaria finita $s^N = s_0, s_1, ..., s_{N-1}, s_N$. Para un polinomio de longitud L $C(D) = 1 + c_1 D + ... + c_L D^L$, sea $\langle L, C(D) \rangle$ un LFSR que genera la subsecuencia $s^n = s_0, s_1, ..., s_{n-1}$. Donde N es la longitud de la secuencia total generada por el LFSR y n es la longitud de una subsecuencia de la secuencia total. La discrepancia d_n es la diferencia entre S_n y el último bit generado por el LFSR [1]:

$$d_n = (s_n + \sum_{i=1}^{L} c_i s_{n-i}) \bmod 2 \qquad (1)$$

La demostración de que este algoritmo es correcto se puede consultar en The Stability Theory of Stream Ciphers C. Ding et al. (1991). EL ABM es un algoritmo iterativo que analiza una secuencia $s^n = s_0, s_1, s_2, \ldots, s_{n-1}$.
En el primer paso se inicializan las variables operativas del programa y las que representan al polinomio de combinación y la complejidad lineal.
La Inicialización de variables se realiza de la siguiente forma:
$C(x) \leftarrow 1$, $L \leftarrow 0$, $m \leftarrow -1$, $B(x) \leftarrow 1$ $k \leftarrow 0$ y $T(x) \leftarrow 0$
Para calcular la discrepancia se usa la ecuación (1). Y para calcular el vector C(x) se hace uso de las variables T(x), B(x), k y m.
$T(x) \leftarrow (x)$; $C(x) \leftarrow C(x) + B(x) \cdot x^{k-m}$.

En la figura 2 se muestra el diagrama de flujo del algoritmo de Berlekamp-Massey. A continuación se muestra un ejemplo de cómo el ABM genera un polinomio a partir de una secuencia de bits con las cuales se alimenta a este algoritmo.

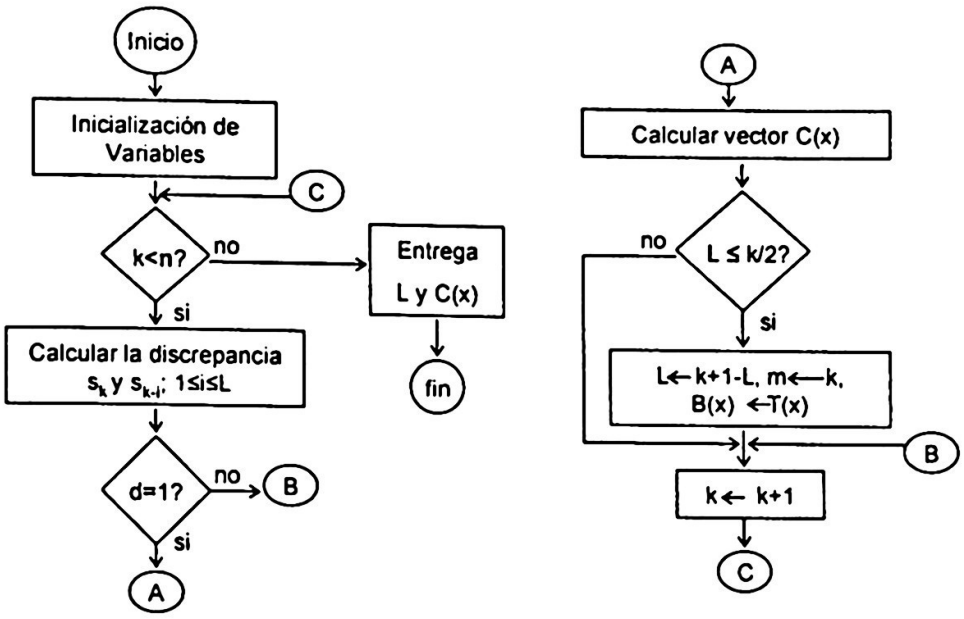

Fig. 2. Diagrama de flujo del algoritmo de Berlekamp – Massey.

Al introducirle al ABM una secuencia de bits como la siguiente: s^n: 011101101001.

El resultado que se obtiene es $P(X)=X^6+X+1$. Este es el polinomio con el que se puede construir un LFSR que consistirá de 6 etapas y se realizará una operación XOR con las salidas de la etapa 6 y 1. Este polinomio es primitivo ya que X^m+1 es divisible solo por P(X), con $m = 2^L-1$ y $L = 6$, pero no lo es cuando $m < 2^L-1$. Por tanto, la secuencia de salida total generada tiene un periodo máximo $(2^L-1) = 63$.

El diseño del Generador Multivelocidad que utiliza LFSR y se trata más adelante ha sido simulado en un FPGA de la familia MAX7000S (EPM7128SLC84-7) con las herramientas que proporciona ALTERA™ y ocupa el 27% de los recursos del chip. Es importante mencionar que no se ocupó ninguna directriz especial de síntesis para optimizar el diseño. Cuando se hace un diseño usando VHDL, herramienta de ALTERA™ le asigna por omisión un retardo a cada uno de los dispositivos que corresponden al FPGA (Field Programmable Gate Array) que se esté utilizando.

A continuación se muestra la secuencia total generada por el LFSR construido a partir del polinomio X^6+X+1 y la semilla 011101, siendo el primer bit de la izquierda más significativo.

Secuencia:011101101001001110001011110010100011000010000011111101010110
011...
Con el ABM y tomando solo 2L bits de cualquier porción de esta secuencia (en este caso L = 6), es decir con solo 12 bits, se puede encontrar que el LFSR que genera esa secuencia tiene el polinomio como ya sabemos X^6+X+1. Esto quiere decir que el cifrado usando la secuencia de salida de un LFSR, no puede asegurar que la información esté protegida, pues es relativamente sencillo obtener esta información usando criptoanálisis si se conocen 2L bits de la secuencia que se usó para cifrarla.

5 Generador Multivelocidad en VHDL

Si combinamos el resultado que ofrece el algoritmo de Berlekamp-Massey con el uso de un lenguaje de descripción de hardware como VHDL (VHSIC Hardware Description Language), se puede desarrollar un programa que genere un LFSR a partir del polinomio que entregue el algoritmo. El diseño de un LFSR en VHDL es muy sencillo, además se puede aprovechar esta herramienta para construir generadores de secuencias pseudoaleatorias que usan combinaciones de LFSR, como el Generador Multivelocidad de este artículo.

Este generador utiliza 2 LFSR que funcionan distintas frecuencias de reloj como se muestra en la figura 3 y el código en VHDL en la tabla 1. El LFSR de L etapas trabaja con un reloj $d \geq 2$ veces más que el de M etapas con $L \geq M$. El factor d es variable y se usa como parte de la clave. Según Rainer A. Rueppel (1986) la diferencia de frecuencia entre los LFSR proporciona flexibilidad y seguridad en aplicaciones criptográficas. Se pueden simular diferentes conexiones de retroalimentación sin cambiar físicamente los estados de retroalimentación, es decir, el polinomio primitivo que indica la retroalimentación puede permanecer constante y lo único que se modifica es la frecuencia, que es más sencillo que cambiar los polinomios de retroalimentación.

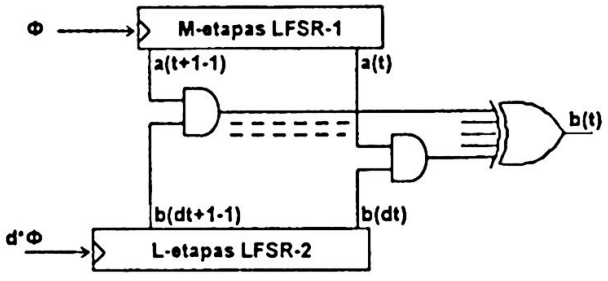

Fig. 3. Generador multivelocidad de Massey-Rueppel.

La incertidumbre acerca de la recursión utilizada en un generador de secuencias pseudoaleatorias hace más difícil el trabajo de un criptoanalista. En particular, la diferencia de frecuencia entre los LFSR disuade al criptoanalista de intentar un ataque de correlación sobre una combinación no lineal de LFSR, pues intentaría buscar a través de todas las recursiones posibles.

Table 1. Código en VHDL del Generador Multivelocidad.

```
library ieee; use ieee.std_logic_1164.all; use ieee.std_logic_arith.all;
library gmr; use gmr.comps_gmr.all;
entity gmr is
generic(M : integer := 7; L:integer:=10; dist : integer := 3; PM : integer := 121;
        INIM : integer := 41;  PL : integer := 911; INIL : integer := 887);
port(clock,ci: in std_logic; salM, salL: out std_logic_vector(M-1 downto 0);
     sal: out std_logic);
end gmr;
Architecture vhdl of gmr is
        signal QA:  std_logic_vector(1 downto 0);
        signal TAP_A,sal_and: std_logic_vector(M-1 downto 0);
        signal TAP_B:  std_logic_vector(L-dist-1 downto 0);
        signal sal_xor:  std_logic;
begin
DIV :    contador port map(clock,QA);
LM:      lfsrm generic map (ancho=>M, polinomio=>PM, inicializacion=>INIM)
         port map(QA(1),TAP_A,ci);
LL:      lfsrl generic map (ancho=>L, polinomio=>PL, inicializacion=>INIL)
         port map(QA(0),TAP_B,ci,QA(1)) ;
AG:      andg generic map(ancho=>M) port map(TAP_A,TAP_B,sal_and);
XORG:  orexg generic map(ancho=>M) port map (sal_and,sal_xor);
salM<=TAP_A;
salL<=TAP_B;
sal<=sal_xor;
end vhdl;
```

5.1 Propiedades del generador multivelocidad

Si los LFSR empleados en el generador se forman con polinomios irreducibles, sus grados L y M son primos relativos y los factores $d_1 \neq d_2$ se escogen tal que:

$$\gcd(d_1, T_1) = \gcd(d_2, T_2) = 1 \tag{2}$$

T_1 y T_2 son los periodos de los LFSR, entonces b(t) tendrá complejidad lineal:

$$CL = LM \tag{3}$$

Con periodo:

$$T \geq \frac{(T1)(T2)}{(q-1)} \qquad ($$

donde q denota el campo de Galois (GF(q)), en este caso q=2 que denota el campo binario [7]. Cuando el generador está restringido a emplear LFSR de longitud L y

(L > M) con polinomios primitivos sobre GF(2), entonces el número de unos y ceros dentro de un periodo $l=(2^M-1)(2^L-1)$ de la secuencia de salida b(t) son:

$$U(1) = (2^M - 1)2^{L-1} \tag{5}$$

$$U(0) = (2^M - 1)(2^{L-1} - 1) \tag{6}$$

y la diferencia de U(1) y U(0) dividida entre la longitud l del periodo es

$$\frac{U(1)-U(0)}{l} = \frac{1}{(2^L-1)} \tag{7}$$

Debe mencionarse que para q>2, los resultados estadísticos no son tan fáciles de obtener [7].

6 Pruebas y Resultados

Se introdujo secuencias de bits al ABM para obtener polinomios que sirven como entrada al código en VHDL para generar los LFSR. A partir de estos, se construye el Generador Multivelocidad. Las secuencias de bits utilizadas son las siguientes:
1) Para el LFSR de M etapas: S = 101010001011110000.
2) Para el LFSR de L etapas: S = 10011101111101110010001.
El ABM proporcionó para el primer caso el siguiente polinomio:
$x^7+x^4+x^3+x^2+x+1$; M = 7. Tiene un periodo de 21 con la siguiente semilla: 1001010.
En el segundo caso, el polinomio obtenido es $x^{10}+x^9+x^8+x^7+x^3+x^2+x+1$; L=10. El periodo de la secuencia de este polinomio es 4 con la siguiente semilla: 1110111011
El código en VHDL para este generador se muestra en la tabla 2 con M = 7 y L = 10 y la simulación se observa en las figuras 4 y 5.
La secuencia de salida del Generador Multivelocidad, usando un factor de frecuencia d=2, es la siguiente. Secuencia: 11111111111111111111111111101111111111111

Este generador proporciona una secuencia con periodo T = 42. El instante en que el generador comienza a entregar los bits de salida es en t = 49.5 ns, y cuando entrega la secuencia completa es en t = 3.41 µs. Como se observa, esta secuencia no es aleatoria, las razones por las cuales este generador no obtuvo el periodo máximo dado por $(2^M-1)(2^L-1)$ y no tiene aleatoriedad se debe a que:

1) los polinomios que se obtuvieron por medio del ABM no son primitivos,
2) la condición $\gcd(d_2, T_2) = 1$, no se cumple debido a que $\gcd(2,4) \neq 1$, y
3) debido a estas razones las ecuaciones (5), (6) y (7), no se cumplen.
Ahora, se construirá el mismo generador pero usando polinomios primitivos.
1) El LFSR de M-etapas es: x^6+x+1 y su semilla es: 011101.
2) El LFSR de L-etapas es: $x^{15}+x+1$ y su semilla es: 110000000000001.

El factor de frecuencia continuará siendo d=2.

Como se están utilizando LFSR con polinomios primitivos se debe cumplir que:
El periodo de la secuencia de salida debe ser: $T = (2^6 - 1)(2^{15} - 1) = 2{,}064{,}321$ bits
El máximo común divisor es: $\gcd(1,63) = 1$ y $\gcd(2,32767) = 1$
y el número de 0's y 1's como se establece en (5) y (6) es:
$U(1) = (2^6 - 1)2^{15-1} = 1{,}032{,}192$; $U(0) = (2^6 - 1)(2^{15-1} - 1) = 1{,}032{,}129$

En las figuras 6 y 7 se muestran el instante de inicio en t= 45.2 ns y final en t=165.1457252 ms de la simulación.

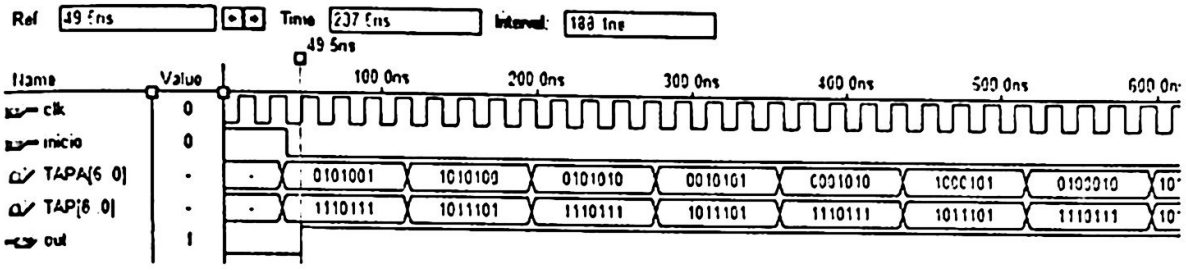

Fig. 4. Inicio de la secuencia pseudoaleatoria en *t = 49.5 ns.*

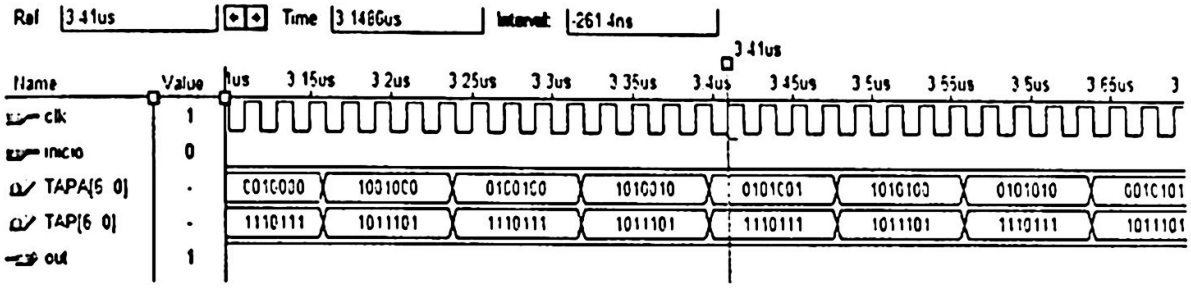

Fig. 5. Final de la secuencia pseudoaleatoria en *t = 3.41μs.*

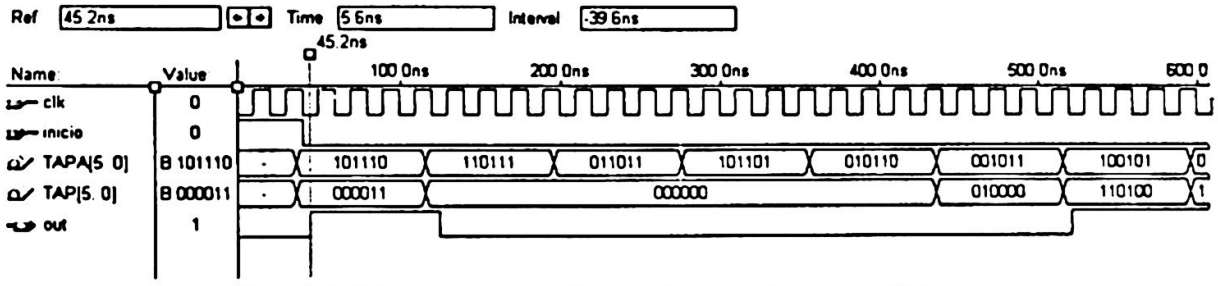

Fig. 6. Inicio de la secuencia pseudoaleatoria en *t = 45.2 ns.*

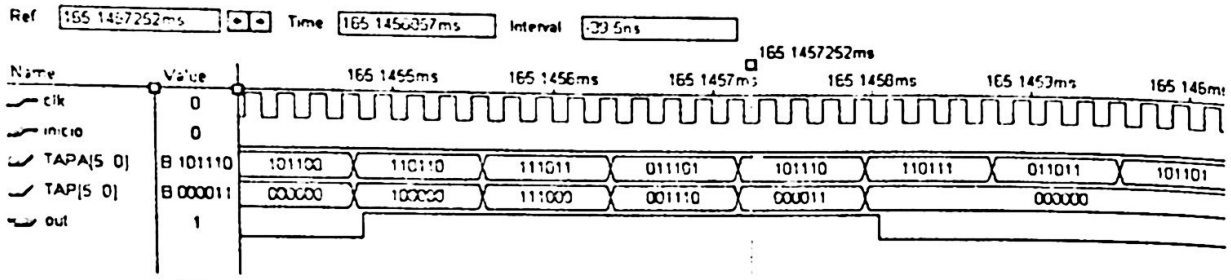

Fig. 7. Final de la secuencia pseudo aleatoria en $t = 165.1457252\ ms$.

7 Conclusiones

Se presentó el algoritmo de Berlekamp-Massey como una herramienta muy eficaz para analizar la seguridad de los LFSR. Se mostró que con 2L bits de una secuencia producida por un LFSR para cifrar información, ésta no se protege, pues se puede encontrar el polinomio con el cual se construye el SLFSR que genera la secuencia con que se cifró la información.

También se presentó el Generador Multivelocidad de Massey-Rueppel. Como el polinomio generado por el ABM no es un polinomio primitivo no se garantiza que pueda ofrecer un periodo máximo al implementarlo en un LFSR. Por tanto, al construir generadores más robustos, como el Generador Multivelocidad con estos polinomios que no cumplen con las propiedades de construcción de este generador en particular, no se podrá obtener una secuencia con periodo máximo.

Al usar polinomios primitivos y cumplir las propiedades del Generador Multivelocidad se pueden obtener secuencias con periodos máximos y buenas propiedades criptográficas. Además haciendo uso del lenguaje de descripción de Hardware como VHDL la construcción de generadores de secuencias pseudoaleatorias se vuelve una tarea sencilla.

Referencias

1. A. Menezes, P. Van, Oorschot, and S. Vanstone: Handbook of Applied Cryptography, CRC Press (1996).
2. A. Fuster, D. de la Guia, L. Hernandez, F. Montoya y J. Muñoz: Técnicas Criptográficas de protección de datos, 2a Edición, Alfaomega (2001).
3. Bruce Schneier, Applied Cryptography: Protocols Algorithms and Source Code in C, Second Edition, John Wiley & Sons, INC (1996).
4. C. Ding, G. Xiao, W. Shan: The Stability Theory of Stream Ciphers; Springer-Verlag Berlin Heidelberg (1991).
5. J. L. Massey y S. Serconek, Linear Complexity of Periodic Sequences: A General Theory, Advances in Cryptology CRYPTO'96, Lecture Notes in Computer Science No. 1109. New York, Springer, pp. 358-371 (1996).
6. Man Young Rhee: Error Correcting Coding Theory, Mc Graw-Hill Communications Series (1989).7. Rainer A. Rueppel: Analysis and Desing of Stream Ciphers, Springer-Verlag (1986).

8. Zeng, K. C.; Yang, C. H. y Rao, T. R.: On the Linear Consistency Test (LTC) in Crypto-analysis with Applications; Proc Crypto'89, Lecture Notes in Computer Science, 435,pp. 164-174 (1989).