GENERACIÓN DE SECUENCIAS BINARIAS PSEUDO ALEATORIAS POR MEDIO DE UN MAPA CAÓTICO 3D.

C.M. González, H. A. Larrondo, C. A. Gayoso, L. J. Arnone

Facultad de Ingeniería. Universidad Nacional de Mar del Plata – Argentina
Juan B. Justo 4302. C. P. 7600. Mar del Plata – Argentina
cmgonzal@fi.mdp.edu.ar

ABSTRACT

This paper deals with both the theoretical and practical realization of a pseudo-random number generator based on a discrete 3D version of a Lorenz chaotic attractor. The practical realization is made on a medium size FPGA. Main differences with other schemes already presented in the open literature are: a) a 3D map is used instead of a 1D map; b) only the least significant byte of one of the state variables is used to generate the primary sequence; c) This sequence is periodically perturbed using another state variable to enlarge the period of the pseudo-random sequence. d) Instead of the floating point numbers required by continuous-value ciphers, natural numbers are used here to simplify the hardware. A comparison with standard number generators as well as the evaluation using standard statistical tests and hardware simulations, are also included.

RESUMEN

Este trabajo presenta el desarrollo teórico y una implementación práctica de un generador de números pseudo aleatorios basado en una versión discreta del sistema dinámico 3D de Lorenz. La realización práctica se implementa en una FPGA de tamaño medio. Las principales diferencias con otros esquemas presentados son las siguientes: a) se utiliza un mapa 3D en lugar de mapas 1D; b) se utiliza sólo el byte menos significativo de una de las variables de estado del sistema para generar la secuencia primaria; c) esta secuencia es periódicamente perturbada usando otra de las variables de estado del sistema para aumentar el período de repetición de la secuencia pseudo aleatoria. d) En lugar de utilizar números en punto flotante, como lo requerirían los sistemas de encriptado continuos, se utilizan números enteros, simplificando el hardware necesario. Se incluye la comparación con generadores standard, los resultados de pruebas estadísticas básicas y la simulación temporal del hardware diseñado.

.

GENERACIÓN DE SECUENCIAS BINARIAS PSEUDO ALEATORIAS POR MEDIO DE UN MAPA CAÓTICO 3D.

C.M. González, H. A. Larrondo, C. A. Gayoso, L. J. Arnone

Facultad de Ingeniería. Universidad Nacional de Mar del Plata – Argentina
Juan B. Justo 4302. C. P. 7600. Mar del Plata – Argentina
cmgonzal@fi.mdp.edu.ar

1.INTRODUCCIÓN

La generación de secuencias aleatorias siempre ha atraído mucho la atención debido a que los números aleatorios juegan un rol muy importante en muchas áreas del conocimiento tales como Ingeniería, Economía, Física, Estadística, etc. En particular son muy utilizados en simulaciones, método de Monte Carlo, criptografía y telecomunicaciones. Toda secuencia producida por una máquina de estados finitos no puede ser realmente aleatoria, debido a que son determinísticas por naturaleza y su salida es predecible. Por lo tanto, por estos medios, se generan secuencias pseudo aleatorias con un período finito, pero si ese período es suficientemente largo la pseudo aleatoriedad es suficiente para muchas de estas aplicaciones[1][2].

En este trabajo se presenta un método de generación de secuencias binarias pseudo aleatorias por medio de la discretización de un sistema caótico 3D, en particular se utiliza la porción menos significativa de las variables de estado como fuente de pseudo aleatoriedad. El sistema se implementa en un dispositivo lógico programable de tamaño medio.

La estructura de este trabajo es la siguiente: en la sección 2 se incluye la discretización del sistema dinámico de Lorenz para obtener el mapa 3D, la modificación para poder implementarlo en aritmética entera y el método perturbativo empleado para alargar el período de la secuencia pseudo aleatoria generada. En el punto 3 se efectúa el análisis estadístico de las secuencias binarias pseudo aleatorias generadas y se comparan los resultados con los obtenidos con otros sistemas conocidos. En el punto 4 se desarrolla la implementación física empleando el dispositivo lógico programable EPF10K20TC144-3 de Altera.

2. GENERACIÓN DE SECUENCIAS CAÓTICAS

El sistema dinámico que se utilizó como punto de partida es un sistema de Lorenz gobernado por las siguientes ecuaciones diferenciales:

$$\begin{cases} \dfrac{dx}{dt} = -\delta(x - y) \\ \dfrac{dy}{dt} = -xz + \Gamma x - y \\ \dfrac{dz}{dt} = xy - bz \end{cases}, \qquad (1)$$

donde δ, Γ y b son los parámetros de control.

El sistema continuo se convierte en un sistema discreto tridimensional (mapa 3D) utilizando la aproximación de Euler de primer orden. El resultado es:

$$\begin{cases} \mathfrak{x}_{n+1} = \mathfrak{x}_n + k\left[-\delta(\mathfrak{x}_n - \mathfrak{y}_n)\right] \\ \mathfrak{y}_{n+1} = \mathfrak{y}_n + k\left[-\mathfrak{x}_n \mathfrak{z}_n + \Gamma \mathfrak{x}_n - \mathfrak{y}_n\right], \\ \mathfrak{z}_{n+1} = \mathfrak{z}_n + k\left[\mathfrak{x}_n \mathfrak{y}_n - b\mathfrak{z}_n\right] \end{cases} \qquad (2)$$

donde k es el parámetro de escala de tiempo. Para reducir el hardware necesario el sistema trabaja con aritmética entera binaria, y para simplificar las divisiones necesarias (se harán desplazando bits) se utilizan múltiplos del tipo 2^n. Para ello se efectúan las siguientes transformaciones de polarización y cambio de escala:

$$\begin{cases} x_n = (\mathfrak{x}_n + B)S \\ y_n = (\mathfrak{y}_n + B)S \\ z_n = (\mathfrak{z}_n + B)S \end{cases}, \qquad (3)$$

donde B y S son respectivamente los parámetros de polarización y cambio de escala. El resultado final es:

$$\begin{cases} x_{n+1} = x_n + k\delta(y_n - x_n) \\ y_{n+1} = (1-k)y_n + (kB + k\Gamma)x_n + \\ \qquad kBz_n - \dfrac{k}{S}x_n z_n + (kBS - k\Gamma BS - kB^2 S) \\ z_{n+1} = (1-kb)z_n - kB(x_n + y_n) + \\ \qquad \dfrac{k}{S}x_n y_n + (kB^2 S + kbBS) \end{cases} \quad ,(4)$$

Los parámetros adoptados en este trabajo son:

$$k = \frac{1}{64}; \quad \delta = 8; \quad \Gamma = 24; \quad b = 2; \quad B = 40; \quad S = 512$$

Estos valores se eligieron para asegurar la estabilidad estructural del sistema (pequeñas modificaciones de los parámetros no alteran el comportamiento cualitativo del sistema), y para facilitar la implementación en aritmética binaria entera. Reemplazando en (4) resulta:

$$\begin{cases} x_{n+1} = x_n + \dfrac{y_n}{8} - \dfrac{x_n}{8} \\ y_{n+1} = y_n - \dfrac{y_n}{64} + x_n + \dfrac{z_n}{2} + \dfrac{z_n}{8} - \\ \qquad \left(\dfrac{x_n}{256}\right)\left(\dfrac{z_n}{128}\right) - 20160 \\ z_{n+1} = z_n - \dfrac{z_n}{32} + x_n - \dfrac{(x_n + y_n)}{2} - \dfrac{(x_n + y_n)}{8} + \\ \qquad \left(\dfrac{x_n}{256}\right)\left(\dfrac{y_n}{128}\right) + 13440 \end{cases} \quad ,(5)$$

Nótese que todas las operaciones involucradas en (5) son sumas, restas y divisiones por potencias de dos, y solamente se utilizan dos multiplicadores.

La figura 1 es una vista 3D de la trayectoria del sistema con condiciones iniciales x_0=18503, y_0=21315 y z_0=32032. Puede verse la típica "mariposa" de Lorenz. La figura 1 también revela que el rango de las variables de estado es [0,6x10^4] y por lo tanto se necesitan al menos 17 bits para representarlas. [3][4][5].

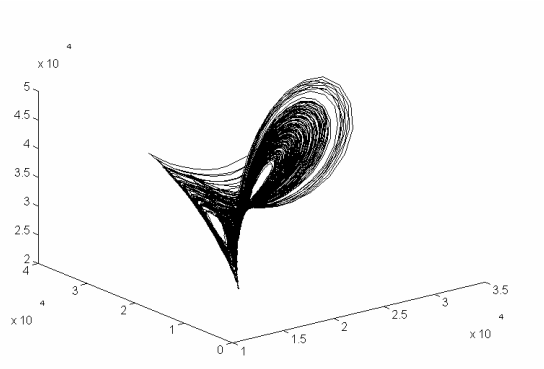

Figura 1. Vista en 3D del atractor de Lorenz

Debido al número finito de dígitos utilizados para representar las variables de estado y al efecto de truncado producido por la división binaria, es inevitable que exista un período de repetición de la secuencia. Por lo tanto el mapa (5) no es caótico sino pseudo caótico.

Incrementando en (4) el factor de escala S es posible incrementar el período de repetición debido a que el efecto del truncado disminuye. Los experimentos numéricos realizados mostraron que el máximo período de repetición que podía obtenerse por este método es 78782. Para muchas aplicaciones este período puede no ser suficientemente largo. En ese caso se lo puede aumentar considerablemente aplicando pequeñas perturbaciones a alguna de las variables de estado. En la figura 2 se muestra una posible estrategia de perturbaciones en la que se modifica la variable X cada **N** ciclos de reloj. Si se utilizara una perturbación de valor fijo (por ejemplo sumar a X un valor constante) la periodicidad no se altería en forma significativa. En cambio si la perturbación es pseudo aleatoria como se muestra en la Figura 2 el período se extiende considerablemente. El método consiste en realizar una OR exclusiva (XOR) entre el byte menos significativo de x_n (x_n[7..0]) y el byte menos significativo de y_n (y_n[7..0]). obteniéndose el byte menos significativo de x'$_n$. El byte más significativo de x'$_n$ coincide con el de x_n . La variable X' se envía a la entrada del sistema de Lorenz. La secuencia pseudo aleatoria final empleada toma sólo el bit menos significativo de la variable X generada.

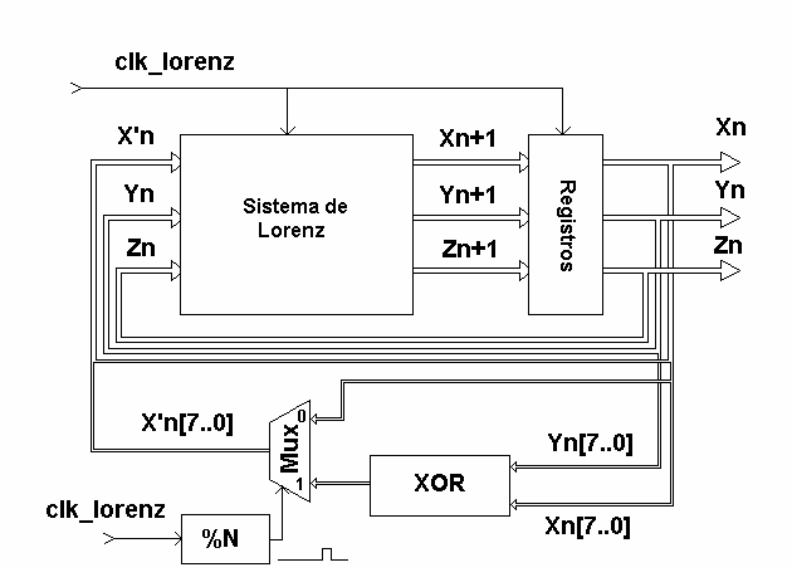

Figura 2. Diagrama en bloques del generador con perturbaciones periódicas.

3. GENERADOR DE SECUENCIAS DE BITS PSEUDO ALEATORIAS.

En las figuras 3a y 3b puede verse que las distribuciones de primer y segundo orden del byte menos significativo de la señal x_n son cualitativamente uniformes. Por otro lado la transformada rápida de Fourier, del byte menos significativo de x_n (figura

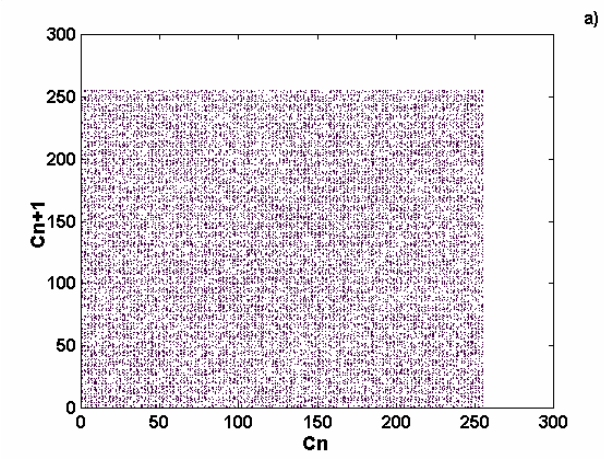

Figura 3. a) Trayectoria 2D para $C_n = X_n[7..0]$ (byte menos significativo de X_n).

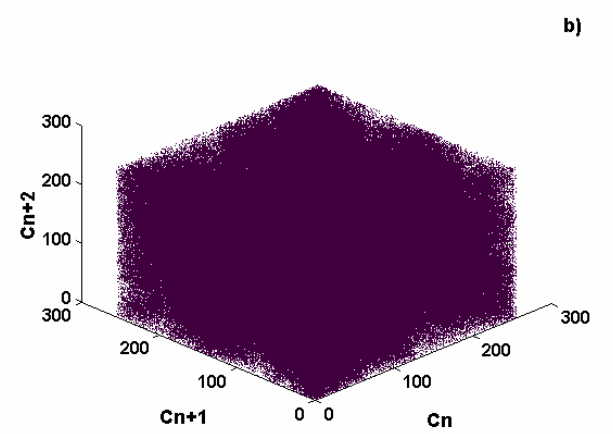

Figura 3. b) Trayectoria 3D.

4) muestra que el espectro es similar al del ruido. Este comportamiento justifica la elección realizada en el diseño pero sólo aporta una visión cualitativa.

Para determinar las características de la secuencia generada y poder compararla con la de otros generadores se realizaron cinco pruebas estadísticas básicas, comúnmente utilizadas (estas pruebas son condiciones necesarias pero no suficientes para demostrar que una secuencia "luzca" aleatoria) [6][7][8].

Las pruebas se efectuaron sobre una secuencia de bits de longitud n=65536: $s = s_0, s_1, s_2, ..., s_{n-1}$. Se realizaron cinco pruebas básicas.

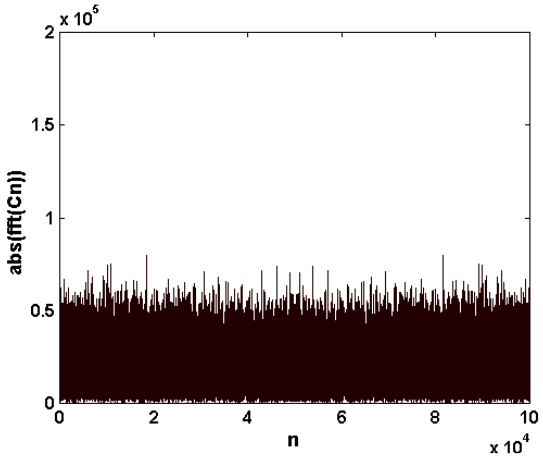

Figura 4. Transformada rápida de Fourier (FFT) de C_n evaluada sobre 100000 muestras.

3.1. Prueba de frecuencia

Esta prueba determina si el n° de 0's (n_0) es aproximadamente igual al n° de 1's (n_1), como debería esperarse si la secuencia fuera aleatoria. Se calcula:

$$X_1 = \frac{(n_0 - n_1)^2}{n} \quad (6)$$

que responde aproximadamente a una distribución χ^2 con un grado de libertad. Los resultados obtenidos son:

n_0 = 32819, n_1 = 32717, n = 65536
X_1 = 0.15875244140625
Tomando α = 0,05 (nivel de significancia)
y ν = 1 (grados de libertad)
Si $X_1 < A(\nu,\alpha) = 3.8415$ (distribución χ^2), entonces se acepta la hipótesis

3.2. Prueba serie

El propósito de esta prueba es determinar si es aproximadamente igual el número de ocurrencias sucesivas de las secuencias 00, 01, 10, 11, como debería esperarse si la secuencia fuera aleatoria. Definimos n_{00} (n° de ocurrencias de la combinación 00), n_{01} (n° de ocurrencias de la combinación 01), n_{10} (n° de ocurrencias de la combinación 10), n_{11} (n° de ocurrencias de la combinación 11), n_o y n_1 como en el caso anterior. Debido a que se permite el solapamiento de las combinaciones, $n_{00} + n_{01} + n_{10} + n_{11} = n - 1$. Se evalúa:

$$X_2 = \frac{4}{n-1}(n_{00}^2 + n_{01}^2 + n_{10}^2 + n_{11}^2) - \frac{2}{n}(n_0^2 + n_1^2) + 1 \quad (7)$$

que sigue aproximadamente una distribución χ^2 con dos grados de libertad. El resultado obtenido es:

n_{00}=16403, n_{01}=16415, n_{10}=16416, n_{11}=16301,
n_0=32819, n_1=32717, n=65536
X_2 = 0.40490056843555
Tomando α = 0,05 y ν = 2
Si $X_2 < A(\nu,\alpha) = 5.9915$, entonces se acepta la hipótesis.

3.3. Prueba del Poker

En esta prueba se separa la secuencia de bits en grupos de ***m*** bits sin solapamiento (Ej. si m=3 [s0 s1 s2] [s3 s4 s5] .....). La secuencia se divide en ***k*** conjuntos de ***m*** bits. Luego se computa el n° de ocurrencias de cada una de las 2^m posibles combinaciones, denominándose $\mathbf{n_i}$, para $1 \le i \le 2^m$. Esta prueba determina si las secuencias de longitud m aparecen aproximadamente el mismo número de veces, como debería esperarse si la secuencia fuera aleatoria. Se computa la siguiente expresión:

$$X_3 = \frac{2^m}{k}\sum_{i=1}^{2^m} n_i^{\,2} - k \quad (8)$$

que sigue aproximadamente una distribución χ^2 con 2^m-1 grados de libertad. Se tomó el bit menos significativo (n=65536) y para el caso m=8 (por lo tanto k=8192) el resultado es:

X_3 = 285.875
Tomando α = 0,05 y ν = 255
Si $X_3 < A(\nu,\alpha) = 293.2478$, entonces se acepta la hipótesis.

3.4. Prueba de rachas

El propósito de esta prueba es determinar si el número de *runs* (rachas de 0's o de 1's) de longitud variable en la secuencia s es el esperado para una secuencia aleatoria. El número esperado de *blocks de 1's* (o *gaps de 0's*) de longitud i en una secuencia aleatoria de longitud n es:

$$e_i = \frac{(n - i + 3)}{2^{i+2}} \quad (9)$$

Sea k el mayor número entero i tal que $e_i \geq 5$ y sean B_i y G_i el número de blocks y de gaps respectivamente, de longitud i con $1 \leq i \leq k$. Se computa la expresión:

$$X_4 = \sum_{i=1}^{k} \frac{(B_i - e_i)^2}{e_i} + \sum_{i=1}^{k} \frac{(G_i - e_i)^2}{e_i} \quad (10)$$

que sigue aproximadamente una distribución χ^2 con 2k-2 grados de libertad. En este caso k=11. Los resultados obtenidos son:

X_4 = 20.31481137118299
Tomando $\alpha = 0{,}05$ y $\nu = 20$
Si $X_4 < A(\nu,\alpha) = 31.4104$, entonces se acepta la hipótesis.

3.5. Autocorrelación

Cálculo de la autocorrelación con desplazamiento d

$$A(d) = \sum_{i=0}^{n-d-1} s_i \oplus s_{i+d} \quad (11)$$

Se calcula:

$$X_5 = \frac{2(A(d) - \frac{n-d}{2})}{\sqrt{n-d}} \quad (12)$$

que sigue aproximadamente una distribución normal N(0,1). n=65536 . Se tomó d=1 y el resultado obtenido es:

X_5 = 0.49609753493826
Tomando $\alpha = 0{,}05$
Si $|X_5| < 1.6449$, entonces se acepta la hipótesis.

Para determinar la calidad de los resultados obtenidos se le realizaron las mismas cinco pruebas a secuencias generadas por tres generadores pseudo aleatorios de amplia utilización:

i) LFSR (Linear Feedback Shift Register) de 34 bits. Este generador se implementa en hardware y se utiliza en comunicaciones.
ii) LCG (Lineal Congruential generator). Algoritmo determinístico para ser utilizado en software, en particular se utilizó el algoritmo de Lehmer usado por la función rand() de Matlab (Z_n=16807*Z_{n-1} mod 2147483647) que genera números entre (0,1). Se lo convirtió a una secuencia de bits (1 si $Z_n >= 0.5$, 0 si $Z_n < 0.5$)
iii) Algoritmo de Marsaglia, determinístico para ser utilizado en software. Z_n=(2111111111*Z_{n-4} + 1492*Z_{n-3} + 1776*Z_{n-2} + 5115*Z_{n-1} + c) mod2^32 c= floor(Z_n /2^32). La secuencia de bits se generó de una manera similar al caso anterior.

En la tabla I se resumen los resultados obtenidos, en todos los casos se utilizaron muestras de 65536 bits.

Tabla I. Cinco pruebas estadísticas básicas sobre muestras de longitud 64Kbits generadas por distintos generadores de secuencias pseudo aleatorias.

Prueba	Valor límite para aceptar la hipótesis (α = 0.05)	Lorenz	LFSR	LCG	Marsaglia
Frecuencia	X_1 < = 3.8415	X_1 = 0.1588	X_1=1.0315	X_1= 1.3010	X_1= 0.0295
Serie	X_2 < = 5.9915	X_2= 0.4049	X_2=1.2476	X_2= 2.9149	X_2= 0.0276
Poker	X_3 < = 293.2478	X_3 = 285.88	X_3=235.44	X_3= 243.13	X_3=266.062
Rachas	X_4 < = 31.4104	X_4 = 20.315	X_4=14.551	X_4= 29.166	X_4= 16.939
Autocorrelación	$\|X_5\|$ < 1.6449	X_5 = 0.4961	X_5=0.4648	X_5= -1.2774	X_5= -0.0273

De los cálculos realizados se desprende que la secuencia evaluada cumple con las cinco pruebas básicas realizadas, siendo sus resultados comparables, desde el punto de vista de estas cinco pruebas básicas, con los obtenidos con los otros métodos. Se están realizando pruebas adicionales más exhaustivas para reafirmar la hipótesis de pseudo aleatoriedad de este generador propuesto, que serán comunicadas en su momento.

4. IMPLEMENTACIÓN EN DISPOSITIVOS LÓGICOS PROGRAMABLES

El generador fue implementado en el circuito integrado EPF10K20TC144-3 de Altera, utilizando el software de desarrollo Max-Plus II.

La figura 5 muestra la arquitectura del diseño realizado, como se ve sigue, en líneas generales, el diagrama en bloques presentado en la figura 2.

La entrada denominada "reloj", es el reloj principal del sistema, a esta señal se la divide por cuatro y se obtiene la frecuencia de generación de cada estado. Las entradas que caracterizan al sistema son: las condiciones iniciales (xin[16..0], yin[16..0] y zin[16..0]) y la señal n[13..0] que da el período de repetición de la perturbación realizada, por lo tanto la "llave" o "semilla" del generador está compuesta por un número de 65 bits. La salida se toma sobre el byte menos significativo de la señal x, xsal[7..0] o directamente se utiliza el bit menos significativo xsal[0]. El bloque denominado DIV_N tiene la función de entregar un pulso, a la selección del multiplexor BUSMUX, cada N períodos de la señal de generación de estados (es decir cada 4N ciclos del reloj principal). El multiplexor mencionado rutea, la salida de XOR (OR exclusiva entre xsal[7..0] e ysal[7..0]) una vez cada N estados, o xsal[7..0] directamente el resto del tiempo en la entrada del bloque lorenz_pn. El bloque lorenz_pn es el encargado de realizar los cálculos necesarios para satisfacer las ecuaciones del sistema, mientras que el bloque REGIS51 realiza el registro y la realimentación de los datos.

Los bloques fueron individualmente implementados en lenguaje VHDL, particularmente en la implementación de las ecuaciones del sistema se utilizaron los Módulos Parametrizados de Biblioteca suministrados por el software Max-Plus II (LPM's), ya que éstos se encuentra optimizados en función del dispositivo utilizado. Las dos multiplicaciones necesarias se implementaron en un único multiplicador en forma secuencial, el resto de las ecuaciones fueron implementadas en paralelo, por lo tanto la limitación en la máxima frecuencia de trabajo será determinada por el retardo de las dos multiplicaciones. Esto se hizo de esta manera para que el sistema pueda ser implementado en el integrado EPF10K20TC144-3 que es el que se disponía, de hecho todo el sistema ocupó 676 celdas lógicas, es decir el 58 % del total de integrado. Queda claro que si se utilizara un integrado con más recursos lógicos se podría duplicar la frecuencia de

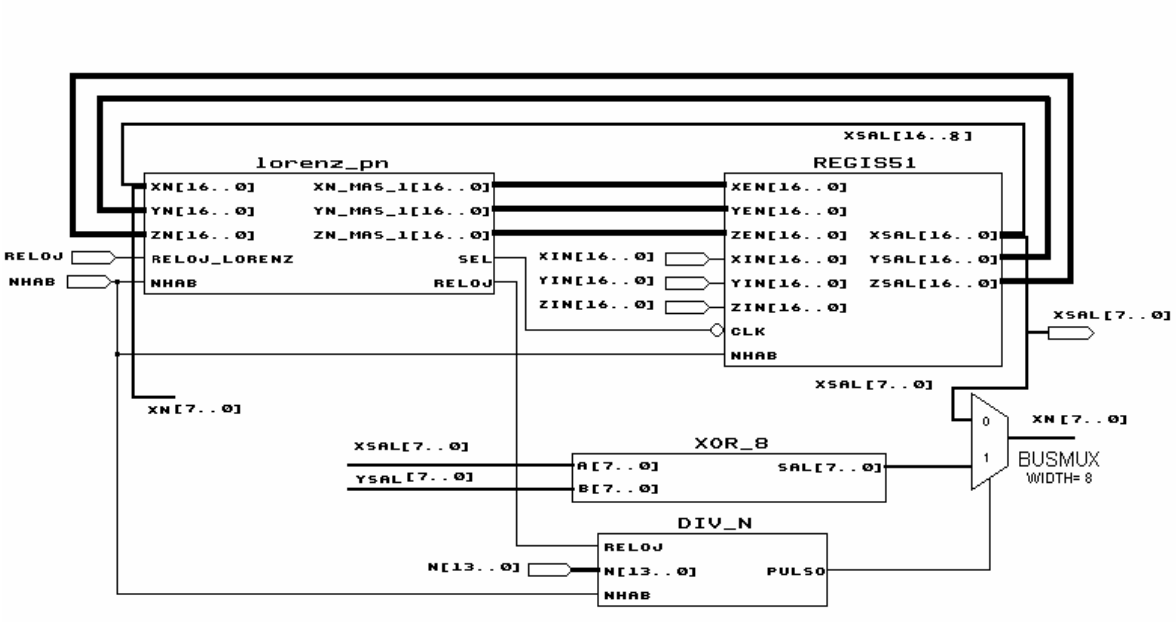

Figura 5. Implementación con Dispositivos Lógicos Programables.

generación de estados, ya que se realizarían las multiplicaciones en paralelo.

Mediante la simulación temporal y el análisis de temporización que realiza el software Max-Plus II se llegó a que la máxima frecuencia de reloj del sistema es de 5.9MHz, por lo que la máxima frecuencia de generación de estados sería de 1.475MHz. Si se toma este diseño como prototipo y se realiza el mismo sistema en un circuito integrado semi o full-custom se podría aumentar considerablemente la frecuencia de trabajo.

5. CONCLUSIONES.

En este trabajo se presentó un generador de secuencias binarias pseudo aleatorias realizado en hardware, obtenido a partir de la discretización de un sistema caótico del tipo de Lorenz. Se parte de la base de que la porción menos significativa de dichas secuencias caóticas tienen una distribución uniforme y un espectro plano. De este modo utilizando el bit menos significativo se genera la secuencia binaria. Se realizaron pruebas básicas de aleatoriedad, por lo que para las aplicaciones más generales puede ser utilizado, si embargo para utilizarlo en aplicaciones más exigentes en donde la seguridad es crítica quedan pendientes pruebas de aleatoriedad que se están realizando actualmente.

En cuanto a su implementación física se realizó en un dispositivo lógico programable de mediano tamaño, con una velocidad de generación de 1.475 Mbits/s, pudiéndose lograr velocidades mucho mayores explotando el cálculo en paralelo de los mapas caóticos, utilizando circuitos integrados semi y full-custom.

5. AGRADECIMIENTOS

Agradecemos el financiamiento de la Universidad Nacional de Mar del Plata, CONICET y ANPCyT.. Agradecemos los comentarios y sugerencias de Eduardo Boemo.

6. BIBLIOGRAFÍA.

[1] R. Bernardini, G. Cortelazzo. “Tools for Designing Chaotic Systems for Secure Random Number Generation”. IEEE Transaction on Circuits and Systems-I, vol. 48, N°48, May 2001, pp. 552-564.

[2] T. Stojanovski, L. Kocarev. “Chaos-Based Random Number Generators-Part. I: Analysis”. IEEE Transaction on Circuits and Systems-I, vol. 48, N°3, March 2001, pp. 281-288.

[3] L. Kocarev, G. Maggio, M. Ogorzalek, L. Pecora, K Yao. (eds). “Special issue on Applications of Chaos in Modern Communication Systems”. IEEE Transaction on Circuits and Systems-I, vol. 48, N°12, 2001.

[4] M. Hasler. “Synchronization of Chaotic Systems and Transmission of Information.” International Journal of Bifurcation and Chaos, vol. 8, N°4, 1998, pp. 647-649.

[5] F. Dachselt, W. Schwarz. “Chaos and Cryptography”. IEEE Transaction on Circuits and Systems-I, vol. 48, N°12, 2001.

[6] A, Papoulis. “Probability, Random Variables, and Stochastic Processes”. Mc Graw-Hill, Inc.. Third Edition, 1991.

[7] A. Menezes, P. Van Oorschot, S. Venstone. “Handbook of Applied Cryptography”. CRC Press. 1997.

[8] U. Maurer. “A Universal Statistical Test for Random Bit Generator”. Journal of Cryptology, vol. 5, N°2, 1992, pp. 89-105.