

FACULTAD DE INVENIERÍA UNIVERSIDAD NACIONAL
DE MAR DEL PLATA

**Sistemas Complejos, Ruidos
Discretos y su implementación en
FPGA**

TESIS

PARA OBTENER EL TÍTULO DE

DOCTOR EN INGENIERÍA CON ORIENTACIÓN EN ELECTRÓNICA

MAXIMILIANO ANTONELLI

DEDICATORIA

Agradecimientos

¡Muchas gracias a todos!

Índice general

1. Introducción	1
2. Cuantificadores de Aleatoriedad	3
2.1. Máximo Exponente de Lyapunov	3
2.1.1. Algoritmo evolutivo para la búsqueda de caos	6
2.2. Cuantificadores de la teoría de la información	14
2.2.1. Entropía de Shannon y Complejidad Estadística	16
2.2.2. Determinación de la distribución de probabilidad	19
2.2.3. Planos doble entropía y entropía-complejidad	23
2.2.4. Cuantificador de entropías implementado en FPGA	29
2.2.5. Dinámica de los ITQ's con AWGN y banda limitada . . .	38
3. Generadores de Números Aleatorios Usando Caos	49
3.1. Sistemas Caóticos	49
3.2. El problema de la Aritmética Discreta	52
3.3. Caos en redes neuronales	52
3.3.1. El modelo de Hopfield	53
3.3.2. Un caso de estudio	55
3.4. Cripto-codificación caótica variante en el tiempo	60
3.4.1. abstract	60
3.4.2. Introducción	60
3.4.3. Mapas cuadráticos bidimensionales	61
3.4.4. Implementación	62
3.4.5. Resultados	66
3.4.6. Conclusiones	67

3.5.	Implementación de atractor Determinístico - Estocástico	68
3.5.1.	Introduction	68
3.5.2.	Discretización temporal del oscilador de Lorenz	71
3.5.3.	Discretización de las variables de estado	72
3.5.4.	Some issues on chaotic PRNG	75
3.5.5.	Results	77
3.5.6.	Conclusions	80
3.6.	Stochastic Degradation of the Fixed-point version of 2D-Chaotic Maps	80
3.6.1.	Introduction	80
3.6.2.	Preliminary Concepts	83
3.6.3.	Results	84
3.6.4.	Conclusion	92
4.	Generadores de TRNG usando ROs en FPGA	95
4.1.	Introduction	95
4.2.	Determinación del <i>jitter</i> en <i>RO's</i>	98
4.3.	Resultados	99
4.4.	Conclusions	108
4.5.	Implementación y análisis estadístico de <i>TRNG</i> basado en <i>ROs</i> .	109
4.5.1.	Resumen	109
4.5.2.	Introducción	109
4.5.3.	Implementación en Hardware	110
4.5.4.	Resultados	114
4.6.	Conclusiones	116
5.	Complejidad de mapas conmutados en precisión finita	119
5.1.	Introduction	119
5.2.	Resultados	122
5.2.1.	Período <i>T</i> en función de <i>B</i>	124
5.2.2.	Cuantificadores de mapas simples	126
5.2.3.	cuantificadores de mapas combinados	130
5.3.	Conclusions	135

6. Conclusiones	141
A. Field Programmable Gate Array (FPGA)	143
Bibliografía	145

Capítulo 1

Introducción

Aqhora arranco con todo...

Capítulo 2

Cuantificadores de Aleatoriedad

2.1. Máximo Exponente de Lyapunov

¿Qué es lo que diferencia a un ciclo límite o una órbita cerrada de una órbita caótica? Una órbita caótica (atractor caótico) es aperiódica, es decir que nunca se repite exactamente y la oscilación persiste para $t \rightarrow \infty$. El movimiento sobre un atrector exhibe una dependencia sensible a las condiciones iniciales. Esto significa que dos trayectorias que comienzan muy cercanas, rápidamente divergen una de otra, por lo que tendrán futuros muy diferentes. La implicación práctica de esto es que la predicción a largo plazo se vuelve imposible en un sistema como este, en donde pequeñas incertezas son amplificadas rápidamente. Hagamos estas ideas un poco más precisas. Supongamos que tenemos una trayectoria sobre el atrlector y un punto $x(t)$ perteneciente a dicha trayectoria en un instante t , ahora consideremos un punto vecino $x(t) + \delta_0$, en donde δ_0 es una pequeña separación inicial. Ahora veamos como evoluciona esta separación $\delta(t)$. Encontramos que

$$\|\delta(t)\| \sim \|\delta_0\| e^{\lambda t} \quad (2.1)$$

Por lo tanto, trayectorias vecinas se separan a un ritmo exponencial. El número λ es llamado exponente de Lyapunov. Cuando este exponente es positivo, se dice que el sistema tiene un horizonte de tiempo t_h más allá del cual la predicción falla por una tolerancia a , de modo que

$$t_h \sim O\left(\frac{1}{\lambda} \ln \frac{a}{\|\delta_0\|}\right) \quad (2.2)$$

Como este sistema presenta un horizonte de tiempo, puede decirse que es sensible a las condiciones iniciales, su exponente de Lyapunov es positivo y resulta ser caótico.

Los exponentes de Lyapunov son quantificadores que caracterizan como evoluciona la separación entre dos trayectorias [?]. En general es bien conocido que el comportamiento caótico está principalmente caracterizado por los números de Lyapunov de la dinámica del sistema.

Venimos llamando al número λ exponente de Lyapunov, sin embargo este es un uso poco riguroso de este término, por dos razones: Primero, λ depende de la trayectoria que estamos estudiando, deberíamos promediar sobre muchos puntos sobre la misma trayectoria para obtener su verdadero valor. Segundo, realmente hay tantos exponentes de Lyapunov como dimensiones tenga el sistema. Supongamos la evolución de una esfera infinitesimal de condiciones iniciales en el espacio de estados de tres dimensiones. Durante esta evolución la esfera se vuelve un elipsoide infinitesimal con tres ejes principales λ_1 , λ_2 y λ_3 , siendo estos tres los exponentes de Lyapunov del sistema. El caos está definido por el máximo exponente de Lyapunov, a partir de ahora MLE, entonces basta que uno de los tres exponentes sea positivo para que el sistema sea caótico. Si uno o más números de Lyapunov es mayor que cero, entonces el sistema se comporta caóticamente, de otra forma el sistema es estable.

El Máximo Exponente de Lyapunov (MLE) caracteriza que tan rápido se apartan dos trayectorias. Si esta velocidad es exponencial, se dice que el sistema es caótico, por lo que este exponente es conocido como un detector de “caotidad”, [?, ?, ?]. Más adelante, el MLE fue utilizado en diversas aplicaciones de muy distintas áreas. Sólo por mencionar alguna, en [?] el MLE es usado para medir una señal muy débil en un gas ideal utilizando criterios caóticos. En [?], se estudia si es posible predecir un cambio en la probabilidad de caída para un

modelo simple de caminante humano a partir del *MLE*.

Sabemos que el sistema en tiempo continuo es una idealización, por lo que se usa el exponente de Lyapunov para tiempo discreto:

$$L = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n \log_2 J_F(x, y, z) \quad (2.3)$$

en donde L es un vector columna de tres dimensiones que contiene los tres exponentes de Lyapunov y $J_F(x, y, z)$ es la matriz jacobiana de la función F para las variables (x, y, z) . El logaritmo es en base dos para estimar la velocidad de apartamiento en bits. Con estas dos consideraciones (tiempo discreto y base numérica binaria finita) la distancia entre dos trayectorias cambia en 2^{MLE} por cada iteración, en promedio. Si el $MLE < 0$ las trayectorias se aproximan, esto puede deberse a un punto fijo. Si el $MLE = 0$ las trayectorias mantienen su distancia, esto puede deberse a un ciclo límite. Si el $MLE > 0$ la distancia entre las trayectorias es creciente, lo que es un indicador de caos.

Calcular L con la ecuación 2.3 tiene dos problemas: Se necesitan infinitas iteraciones para hallar los exponentes de Lyapunov. Trabajar con el jacobiano puede ser computacionalmente pesado, o éste puede no existir. Afortunadamente existe un algoritmo no analítico por aproximaciones sucesivas que converge al máximo exponente de Lyapunov. Las entradas y las salidas de un sistema deben ser accesibles para poder utilizarlo. El procedimiento es el siguiente: el sistema debe ser iniciado desde dos puntos cercanos en el plano de fase, llamémoslos (x_a, y_a) y (x_b, y_b) . A medida que el sistema es iterado se mide la distancia euclídea entre las dos trayectorias (d_n en la muestra n_{th}) (eq. 2.4), y la trayectoria b es relocalizada en cada iteración (eq. 2.6) obteniendo los puntos (x_{br}, y_{br}) para realimentar el sistema. Entonces, el MLE puede ser calculado como se muestra en la ecuación 2.5. El proceso puede verse en la Fig. 2.1.

$$\begin{aligned} d_{0(i-1)} &= \sqrt{(x_{a(i-1)} - x_{br(i-1)})^2 + (y_{a(i-1)} - y_{br(i-1)})^2} \\ d_{1(i)} &= \sqrt{(x_{a(i)} - x_{b(i)})^2 + (y_{a(i)} - y_{b(i)})^2} \end{aligned} \quad (2.4)$$

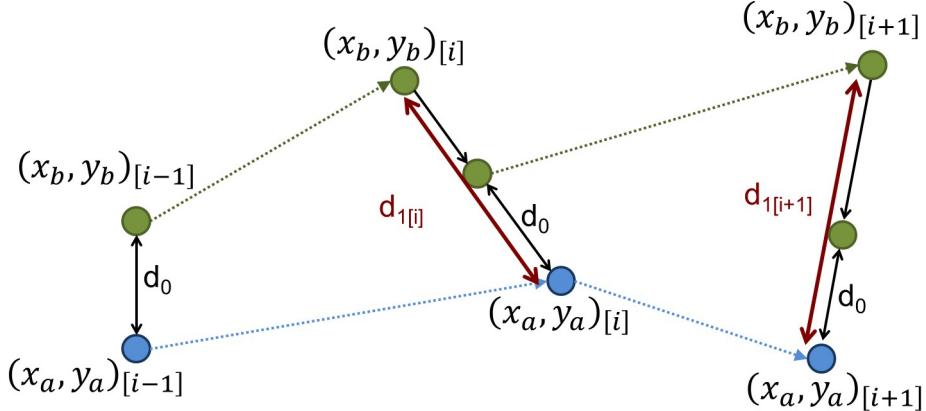


Figura 2.1: Algoritmo para calcular el MLE.

$$MLE = \frac{1}{n} \sum_{i=2}^n \log_2 \frac{d_{1(i)}}{d_{0(i-1)}} \quad (2.5)$$

$$\begin{aligned} x_{br(i)} &= x_{a(i)} + (x_{b(i)} - x_{a(i)})d_{o(i-1)}/d_{1(i)} \\ y_{br(i)} &= y_{a(i)} + (y_{b(i)} - y_{a(i)})d_{o(i-1)}/d_{1(i)} \end{aligned} \quad (2.6)$$

2.1.1. Algoritmo evolutivo para la búsqueda de caos

Propusimos emplear un método eurístico para buscar parámetros del sistema implementado de tal forma que se maximice la caoticidad de su salida. Nuestro trabajo tiene la ventaja que realiza una búsqueda inteligente mediante el empleo de un algoritmo genético, lo que minimiza el tiempo de cómputo.

Un algoritmo evolutivo es un método de búsqueda dirigido basado en la probabilidad. Un juego de entidades que representan posibles soluciones compite con otros, evolucionando en mejores soluciones [?].

Las entidades que representan posibles soluciones al problema son llamados *cromosomas* y el grupo de cromosomas es llamados *población inicial*.

Desde la población inicial, o los primeros padres, se genera un hijo mediante el cruce entre ellos. Luego, ellos son mutados en forma aleatoria para crear la

próxima generación. Cada generación es comparada con la previa para descartar los “peor adaptados” así los coeficientes (cromosomas) mutan hacia los “mejor adaptados”.

Cuando se aplican estos algoritmos en funciones continuas, siempre convergen hacia el máximo local. Sin embargo, si el espacio de coeficientes es fractal, existen áreas bien definidas en donde el la función objetivo es positiva, negativa, cero o no existente. Este es el caso si la función a maximizar es el MLE y el espacio de exploración es el de parámetros.

Resultados

Para evaluar la viabilidad del método, se generó el siguiente algoritmo y se probó sobre el mapa logístico.

En la figura ?? podemos ver el diagrama de flujo principal. El bloque *Evolution* fue descompuesto en otro sub-diagrama para simplificar la descripción. Este segundo diagrama puede verse en la figura ??, esta surutina maneja la evolución de los parámetros.

El algoritmo inicia con una inicialización general de parámetros como el número máximo de generaciones *max_gen*, el número máximo de mutaciones *max_mut* y el número máximo de cambios en cada mutación *max_stem*. Luego se definen los primeros dos padres, ellos definirán los márgenes de búsqueda. Además se calcula su *fitness function Fp*. A partir de este punto se itera la segunda generación, se elige en forma aleatoria un valor de parámetro *r* con una distribución aleatoria entre los primeros dos padres, generando un nuevo hijo. Luego este hijo entra en la subrutina *Evolution* cuya salida es el valor de *r* evolucionado y su correspondiente *Fc*.

Luego se evalúa si este hijo avolucionó muy cerca de sus padres o no. Si la distancia entre ellos es más grande que el parámetro *max_hop*, entonces este hijo es considerado como adulto, en caso contrario debe competir con su padre más cercano sobreviviendo el más apto.

Este proceso se repite hasta que se llega al máximo número de generaciones *max_gen*. El grupo final de adultos es la solución al problema de buscar los máximos MLE locales.

La subrutina *Evolution* de la figura 2.3) es un algoritmo muy simple basado

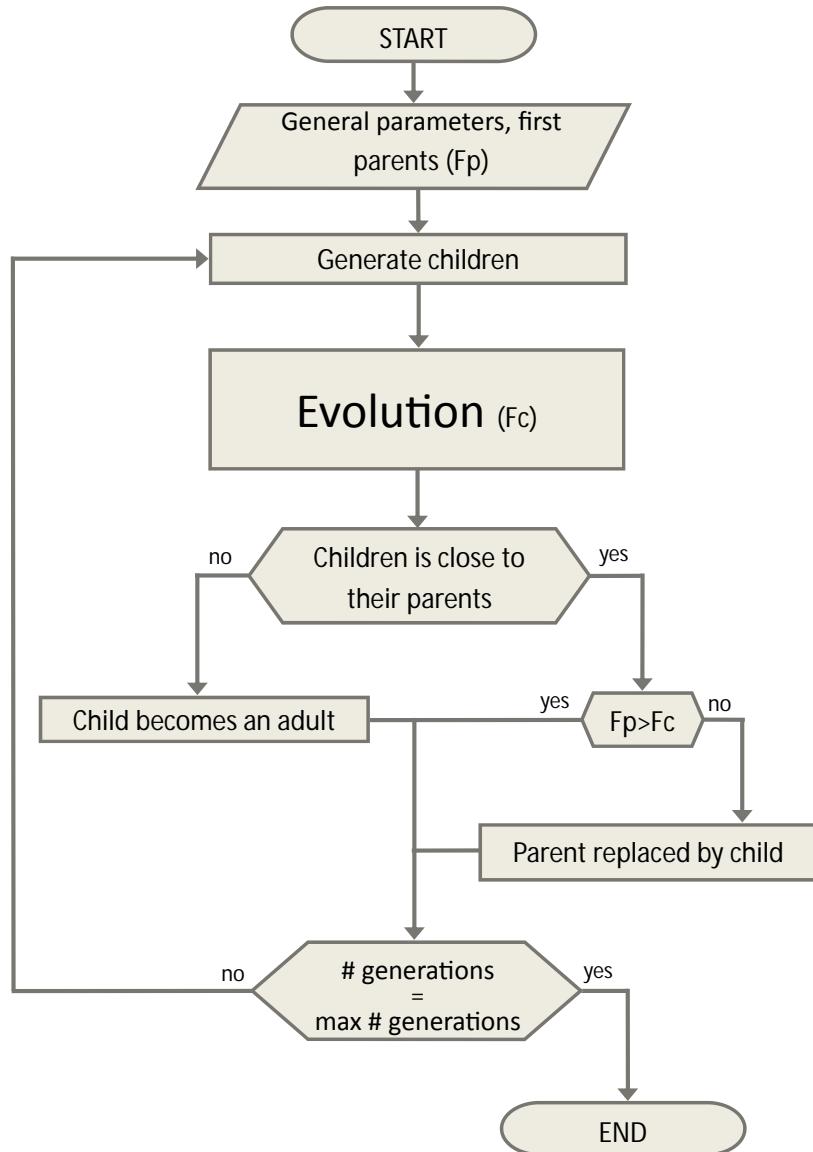


Figura 2.2: Main flow chart.

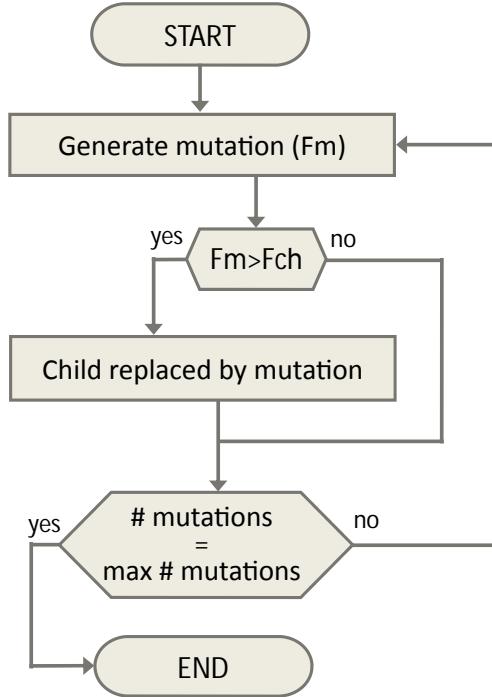


Figura 2.3: Evolution flow chart.

en mutaciones. El primer paso es generar una mutación del hijo con una probabilidad uniformemente distribuida entre $\pm max_step$, tambien se calcula su *fitness function* Fm , que se compara con la del individuo original Fc . Entonces sobrevive el mejor adaptado para dar lugar a la siguiente mutación. Este procedimiento se repite hasta que se llega al máximo número de mutaciones max_mut .

Como resultado podemos ver el *MLE* del mapa logístico en función de su único parámetro r en la figura 2.4. La línea continua muestra el *MLE* en pasos continuos de r , mientras que los puntos destacados son el resultado del algoritmo propuesto.

El bloque que calcula el *MLE* fue sintetizado y verificado experimentalmente en un Altera CYCLONE III FPGA y los resultados de la compilación mostrados en la figura 2.5. Los resultados del *Timing Analysis* reportan que la máxima frecuencia es de $84,95MHz$. El reporte de compilación muestra que la utilización de la lógica no excede el 20 %, es decir un total de 20307 de elementos lógicos, 54 % de los bits de memoria totales y 8 % de los multiplicadores embebidos.

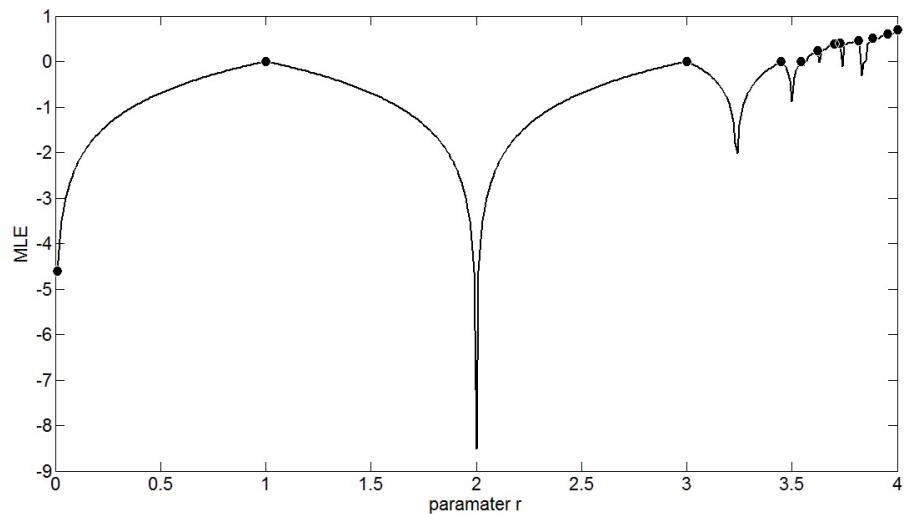


Figura 2.4: Algorithm results.

Flow Summary	
Flow Status	Successful - Fri Apr 19 10:20:17 2013
Quartus II 32-bit Version	12.1 Build 177 11/07/2012 SJ Web Edition
Revision Name	CalculaLyap
Top-level Entity Name	TOP
Family	Cyclone III
Device	EP3C120F780C7
Timing Models	Final
Total logic elements	29,307 / 119,088 (25 %)
Total combinational functions	26,048 / 119,088 (22 %)
Dedicated logic registers	18,014 / 119,088 (15 %)
Total registers	18014
Total pins	197 / 532 (37 %)
Total virtual pins	0
Total memory bits	2,133,356 / 3,981,312 (54 %)
Embedded Multiplier 9-bit elements	48 / 576 (8 %)
Total PLLs	1 / 4 (25 %)

Figura 2.5: Compilation report of the *MLE* calculator.

Type	Alias	Name	-4085	salut	4084	-4096	-3584	-3072	-2560	-2048	-1536	-1024	-512	0	512	
out		salida	1171429278		1171430150	1171430139	1171430659	1171431953	1171431895	1171433747	1171432048	1171434276	1171435581			
out		cuenta_sal	95475		95475	95476	95477	95478	95479	95480	95481	95482	95483	95484		
out		listoD1	1													

Figura 2.6: Signal Tap output.

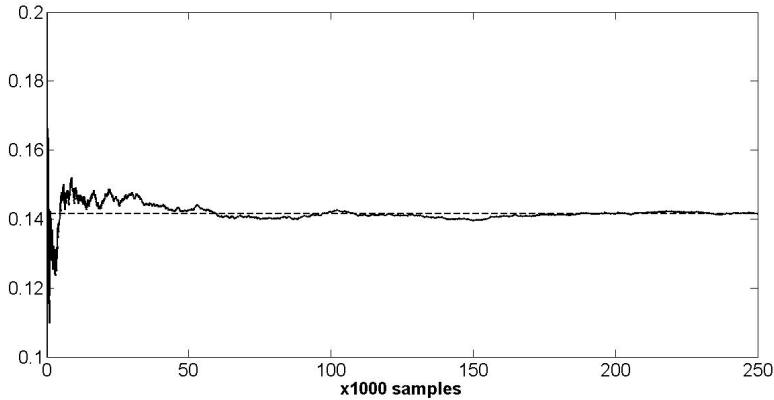


Figura 2.7: Lyapunov curve

En la figura 2.6 se muestra la salida del Signal Tap. La señal *salida* es la suma de los *MLE* luego de cada iteración. La segunda señal llamada *cuenta_sal* corresponde a la sumatoria actual. Finalmente, cada flanco descendente de la señal *listoD1* indica que la salida es un dato válido. La salida fué procesada con Matlab para obtener la curva mostrada en la figura 2.7. El valor del *MLE* en la iteración 250000 es 0,1415, lo que es consistente con el *MLE* obtenido con Matlab.

Estado actual del avance

Actualmente estamos en etapa de desarrollo de la implementación en hardware de este algoritmo. En este segundo caso, el sistema bajo prueba es la familia de mapas cuadráticos bidimensionales descritos en el capítulo ?? y cuya ecuación es ??.

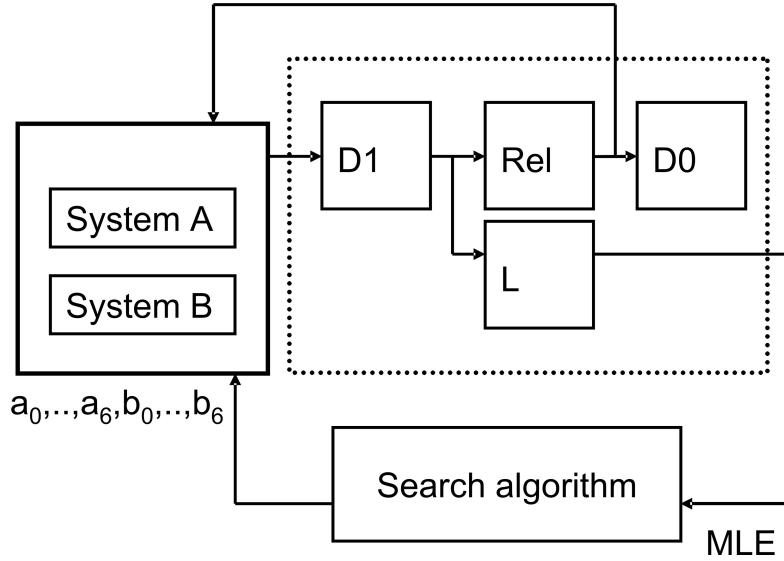


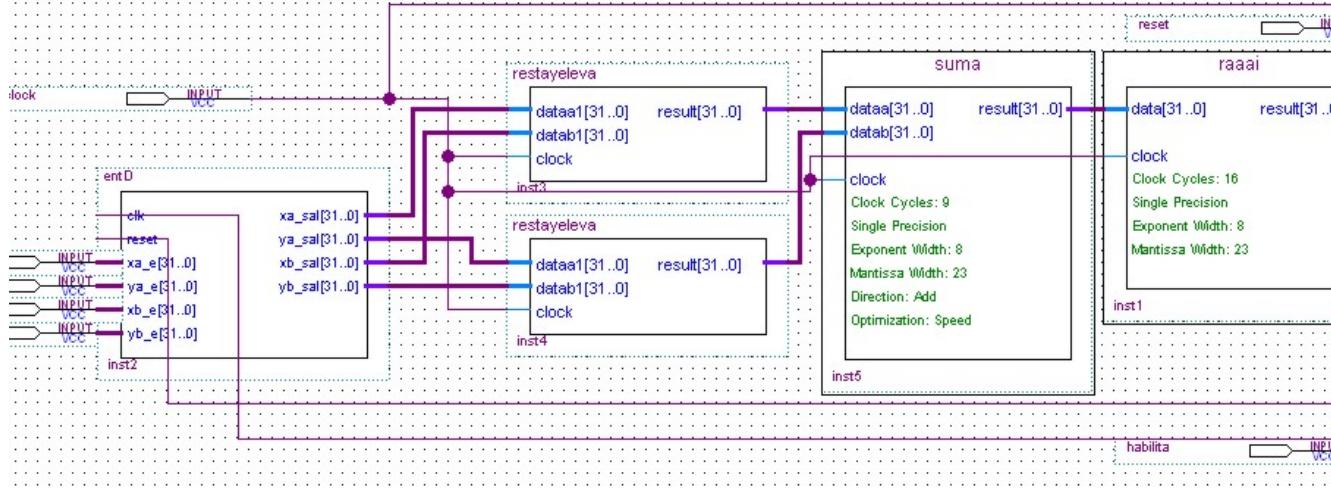
Figura 2.8: Enabling flow of the System.

$$\begin{aligned}
 x_{(i+1)} = & a_0 + a_1 x_{(i)} + a_2 x_{(i)}^2 + \\
 & + a_3 x_{(i)} y_{(i)} + a_4 y_{(i)}^2 + a_5 y_{(i)} \\
 y_{(i+1)} = & b_0 + b_1 x_{(i)} + b_2 x_{(i)}^2 + \\
 & + b_3 x_{(i)} y_{(i)} + b_4 y_{(i)}^2 + b_5 y_{(i)}
 \end{aligned} \tag{2.7}$$

La población inicial es de 12 coeficientes iniciales del mapa caótico empleado. En la figura 2.8 puede verse un diagrama en bloques general del sistema. Este consiste en dos bloques principales conectados al sistema caótico bajo prueba a través de una interface wishbone. Esto independiza el sistema del cuantificador y permite cambiar fácilmente el sistema bajo prueba.

El sistema caótico es duplicado en dos bloques, *SystemA* y *SystemB*. Cada uno de ellos es inicializado con los puntos en el espacio de fases $(x_{a(i-1)}, y_{a(i-1)})$ y $(x_{br(i-1)}, y_{br(i-1)})$ respectivamente. Cuando los sistemas caóticos terminan de calcular sus salidas, la señal digital *habilita* se pone en cero y el bloque *D1* es habilitado para calcular la distancia euclídea entre las salidas $(x_{a(i)}, y_{a(i)})$ y $(x_{br(i)}, y_{br(i)})$.

Luego se habilitan los bloques concurrentes *L* y *Rel*. Los puntos relocaliza-

Figura 2.9: D_1 Block.

dos que alimentan al bloque *SystemB* son calculados por el bloque *Rel*. Este bloque solo necesita los valores actuales de d_1 los valores previos de d_0 , como se muestra en la ecuación 2.6. Cuando los puntos relocalizados $x_{br(i)}$ y $y_{br(i)}$ están disponibles, los bloques *SystemA* y *SystemB* se habilitan para obtener la siguiente iteración. También se habilita el bloque D_0 para calcular el valor actual de $d_{0(i)}$, que se utilizará en la siguiente iteración.

Finalmente, el bloque *L* realiza la división entre d_0 y d_1 para luego calcular el valor absoluto y el logaritmo de esta división. El mapa es iterado $N = 250000$ veces y el resultado de la sumatoria dividido por N para asegurar la convergencia del método.

Cada bloque fue implementado utilizando lenguaje VHD e IP *cores* provistos por *Altera* (*megafuctions*) cada vez que fue posible, debido a que estos *cores* están optimizados para este dispositivo. Las operaciones de punto flotante como las sumas, multiplicaciones, valores absolutos y logaritmos fueron calculadas con dichas *megafuctions*.

La figura 2.9 muestra la implementación del bloque D_1 en el entorno gráfico Quartus. Los puntos de salida y entrada a y b , son tomados luego de que la señal *habilita* se pone en cero. Entonces, las señales son procesadas de acuerdo a la ecuación 2.4 para calcular la distancia euclídea d_1 .

La lógica del algoritmo genético fue implementada en lenguaje VHD. Esta lógica junto con el registro de los 12 coeficientes se muestran en la figura 2.10.

También se muestran los resultados de la compilación en la figura 2.11. Puede verse que esta implementación ocupa pocos recursos del dispositivo.

2.2. Cuantificadores de la teoría de la información

Los sistemas dinámicos son sistemas que evolucionan en el tiempo. En la práctica, solo es posible medir una serie de tiempo escalar $X(t)$ la cual puede ser función de las variables $V = \{v_1, v_2, \dots, v_k\}$ que describe la dinámica subyacente (por ejemplo $dV/dt = f(V)$). Tratamos de inferir propiedades de un sistema no conocido a partir del análisis de los datos guardados de datos observacionales. ¿Cuanta información revelan estos datos sobre la dinámica del sistema o procesos subyacentes?.

El contenido de información de un sistema se evalúa típicamente mediante una función de distribución de probabilidad (PDF) P que describe la distribución de alguna cantidad mensurable o observable, generalmente una serie de tiempo $X(t)$. Podemos definir los cuantificadores de la Teoría de la Información como medidas capaces de caracterizar las propiedades relevantes de las PDFs asociadas a estas series temporales, y de esta manera debemos extraer juiciosamente información sobre el sistema dinámico en estudio. Estos cuantificadores representan métricas en el espacio de PDFs para conjuntos de datos, permitiendo comparar diferentes conjuntos y clasificarlos de acuerdo a sus propiedades de procesos subyacentes, de manera amplia, estocástica vs. determinística.

En nuestro caso, nos interesa la dinámica caótica. Por lo tanto, nos centramos en las métricas que toman en cuenta el orden temporal de las observaciones de forma explícita; es decir, el enfoque es fundamentalmente de naturaleza *causal* y *estadística* en la naturaleza. En un enfoque puramente estadístico, las correlaciones entre los valores sucesivos de las series temporales se ignoran o simplemente se destruyen a través de la construcción del PDF; mientras que un enfoque causal se centra en las PDFs de secuencias de datos.

Los cuantificadores seleccionados se basan en el recuento de símbolos y en la estadística de patrones de orden. Las métricas a utilizar pueden clasificarse de forma amplia en dos categorías: las que cuantifican el *contenido de información*

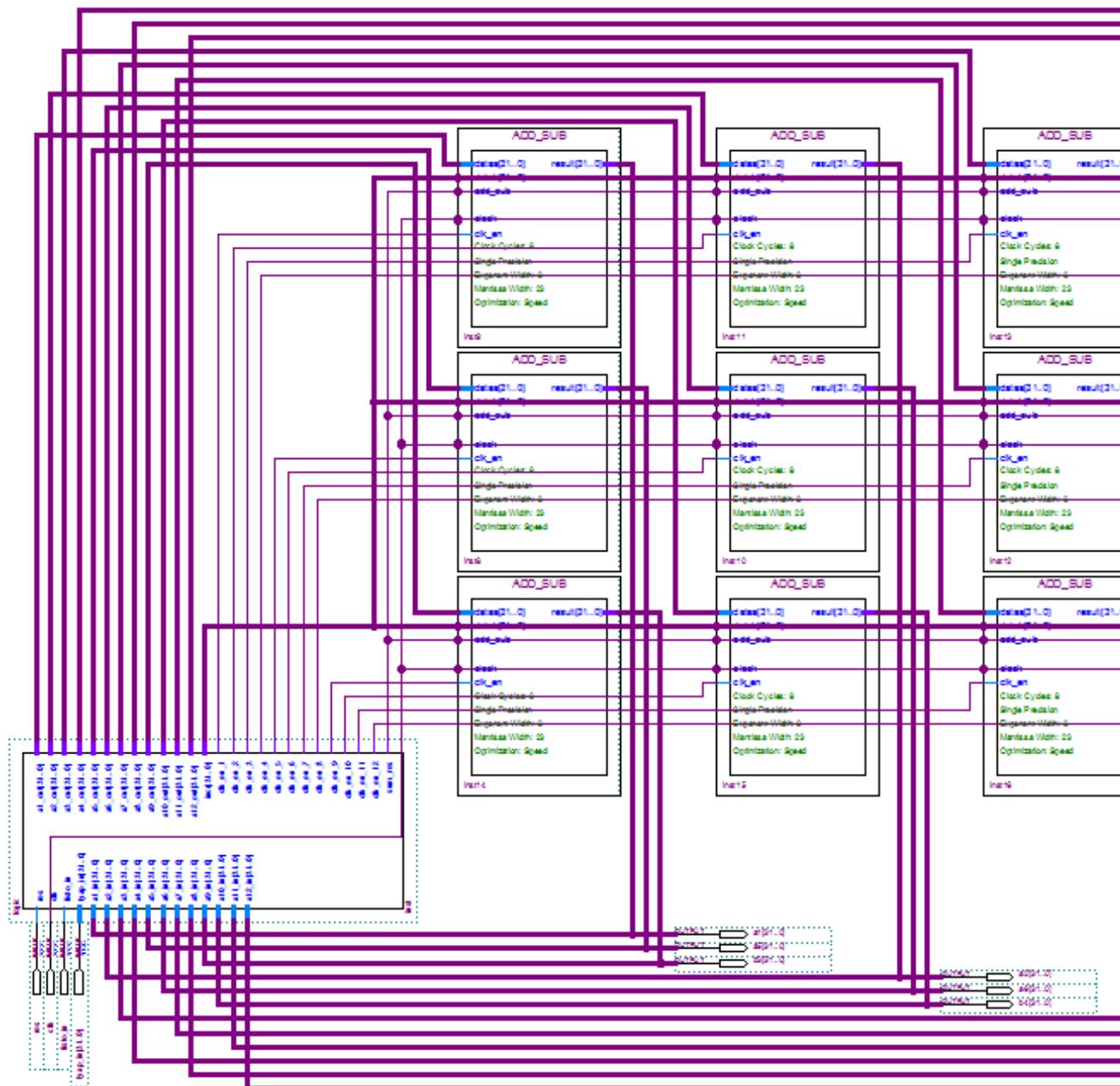


Figura 2.10: Circuit of the evolutive algorithm.

Flow Summary	
Flow Status	Successful - Mon Apr 22 11:03:28 2013
Quartus II 64-Bit Version	12.1 Build 243 01/31/2013 SP 1 SJ Web Edition
Revision Name	feedback
Top-level Entity Name	feedback
Family	Cyclone III
Device	EP3C120F780C7
Timing Models	Final
Total logic elements	10,136 / 119,088 (9 %)
Total combinational functions	9,700 / 119,088 (8 %)
Dedicated logic registers	5,093 / 119,088 (4 %)
Total registers	5093
Total pins	419 / 532 (79 %)
Total virtual pins	0
Total memory bits	432 / 3,981,312 (< 1 %)
Embedded Multiplier 9-bit elements	0 / 576 (0 %)
Total PLLs	0 / 4 (0 %)

Figura 2.11: Compilation report of the evolutive algorithm.

mación de los datos en comparación con los relacionados con su *complejidad*. Obsérvese que aquí nos estamos refiriendo al espacio de funciones de densidad de probabilidad, no al espacio físico. Para clarificar y simplificar, introducimos solamente los cuantificadores de la Teoría de la Información que se definen en PDFs discretas, ya que solo estamos tratando con datos discretos (series temporales). Sin embargo, todos los cuantificadores también tienen definiciones para el caso continuo [?].

2.2.1. Entropía de Shannon y Complejidad Estadística

La entropía es una cantidad básica que puede considerarse como una medida de la incertidumbre asociada (información) al proceso físico descrito por P . Al tratar con el contenido de la información, la entropía de Shannon se considera a menudo como la fundamental y más natural [?]. Considerada como una medida de la incertidumbre, es el ejemplo más paradigmático de estos cuantificadores de información.

Sea $P = \{p_i; i = 1, \dots, N\}$ con $\sum_{i=1}^N p_i = 1$, una distribución de probabilidad discreta, con N el número de estados posibles del sistema bajo estudio. La

medida de la información logarítmica de shannon se denota como

$$S[P] = - \sum_{i=1}^N p_i \ln [p_i] . \quad (2.8)$$

Si $S[P] = S_{\min} = 0$, estaremos en posición de predecir con total certeza cuáles de los posibles resultados i , cuyas probabilidades están dadas por p_i , tendrán lugar realmente. Nuestro conocimiento del proceso subyacente descrito por la distribución de probabilidad es máximo en este caso. Por el contrario, nuestro conocimiento es mínimo para una distribución uniforme $P_e = \{p_i = 1/N; i = 1, \dots, N\}$ dado que cada resultado exhibe la misma probabilidad de ocurrencia, y la incertidumbre es máxima, es decir, $S[P_e] = S_{\max} = \ln N$. Estas dos situaciones son casos extremos, por lo tanto nos centramos en la entropía de Shannon "normalizada", $0 \leq H \leq 1$, dada como

$$H[P] = S[P]/S_{\max} . \quad (2.9)$$

Contrariamente al contenido de la información, no existe una definición universalmente aceptada de complejidad. Aquí, nos centramos en describir la *complejidad de las series temporales* y no nos referimos a la complejidad de los *sistemas* subyacentes. Un sistema complejo no genera necesariamente una salida compleja. De hecho, los modelos "simples" pueden generar datos complejos, mientras que los sistemas "complicados" pueden producir datos de salida de baja complejidad [?].

Una noción intuitiva de una complejidad cuantitativa atribuye valores bajos tanto a datos perfectamente ordenados (es decir, con entropía de Shannon que se va desapareciendo) como a datos aleatorios no correlacionados (con entropía Shannon máxima). Por ejemplo, la complejidad estadística de una simple oscilación o tendencia (ordenada), pero también de ruido blanco no correlacionado (no ordenado) sería clasificada como baja. Entre los dos casos de mínima y máxima entropía, los datos son más difíciles de caracterizar y por lo tanto la complejidad debe ser mayor. Buscamos alguna función $C[P]$ que cuantifique las estructuras presentes en los datos que se alejan de estos dos casos. Estas estructuras se relacionan con la organización, la estructura correlacional, la memoria,

la regularidad, la simetría, los patrones y otras propiedades [?].

Asumimos que el grado de estructuras correlacionales sería capturado adecuadamente por algún funcional $C[P]$ de la misma manera que la entropía de Shannon $S[P]$ [?] “captura” la aleatoriedad. Claramente, las estructuras ordinales presentes en un proceso no son cuantificadas por medidas de aleatoriedad y, por consiguiente, son necesarias medidas de complejidad estadística o estructural para una mejor comprensión (caracterización) de la dinámica del sistema representada por sus series temporales [?].

Una medida adecuada de complejidad puede definirse como el producto de una medida de información y una medida de desequilibrio, es decir, algún tipo de distancia de la distribución equiprobable de los estados accesibles de un sistema. En este sentido, en [?] los autores introdujeron una eficaz *Medida de Complejidad Estadística* (SCM) C , que es capaz de detectar detalles esenciales de los procesos dinámicos subyacentes al conjunto de datos. Basado en el trabajo de López-Ruiz [?], esta medida de complejidad estadística [?, ?] se define a través de la forma del producto

$$C[P] = Q_J[P, P_e] \cdot H[P] \quad (2.10)$$

de la entropía de Shannon normalizada H , ver eq. (2.9), y el desequilibrio Q_J definido en términos de la divergencia de Jensen-Shannon $J[P, P_e]$. Esto es,

$$Q_J[P, P_e] = Q_0 J[P, P_e] = Q_0 \{S[(P + P_e)/2] - S[P]/2 - S[P_e]/2\}, \quad (2.11)$$

en la divergencia de Jensen-Shannon mencionada arriba, Q_0 es una constante de normalización tal que $0 \leq Q_J \leq 1$:

$$Q_0 = -2 \left\{ \frac{N+1}{N} \ln(N+1) - \ln(2N) + \ln N \right\}^{-1}, \quad (2.12)$$

y es igual a la inversa del máximo valor posible de $J[P, P_e]$. Este valor es obtenido cuando una de las componentes de P , digamos p_m , es igual a uno y todos los p_j restantes son cero.

La divergencia de Jensen-Shannon, que cuantifica la diferencia entre las distribuciones de probabilidad, es especialmente útil para comparar la composición

simbólica entre diferentes secuencias [?]. Obsérvese que la SCM introducida anteriormente depende de dos distribuciones de probabilidad diferentes: una asociada con el sistema analizado, P , y la otra con la distribución uniforme, P_e . Además, se demostró que para un valor dado de H , el rango de valores posibles de C varía entre un mínimo C_{min} y un máximo C_{max} , restringiendo los posibles valores del SCM [?].

Por lo tanto, está claro que información adicional importante relacionada con la estructura correlacional entre los componentes del sistema físico se proporciona evaluando la medida de la complejidad estadística.

2.2.2. Determinación de la distribución de probabilidad

La evaluación de los cuantificadores derivados de la Teoría de la Información supone algún conocimiento previo sobre el sistema; específicamente para aquellos introducidos previamente (entropía de Shannon y complejidad estadística), una distribución de probabilidad asociada a la serie temporal en análisis debe proporcionarse antes. La determinación del PDF más adecuado es un problema fundamental porque la PDF P y el espacio de muestra Ω están intrincadamente vinculados.

Las metodologías usuales asignan a cada valor de la serie $X(t)$ (o conjunto de valores consecutivos no superpuestos) un símbolo de un alfabeto finito $A = \{a_1, \dots, a_M\}$, creando así una *secuencia simbólica* que puede considerarse como una descripción de la serie cronológica en cuestión. Como consecuencia, las relaciones de orden y las escalas temporales de la dinámica se pierden por completo.

Es importante resaltar que P en si, no es un objeto con una definición única y existen varias aproximaciones para “asociar” una dada P con una dada serie de tiempo. Solo para mencionar algunos criterios de extracción utilizados frecuentemente en la literatura: *a*) histogramas de series temporales [?], *b*) dinámica simbólica binaria [?], *c*) análisis de Fourier [?], *d*) transformadas wavelet [?, ?], *e*) PDF de particiones [?], *f*) PDF de permutaciones [?, ?], *g*) PDF discreta [?], etc. Hay una amplia libertad para elegir entre ellas y la aplicación específica debe ser analizada para hacer una buena elección.

Se puede incorporar debidamente la información causal si se incluye infor-

mación sobre la dinámica pasada del sistema en la secuencia simbólica, es decir, los símbolos del alfabeto A se asignan a una porción del espacio de fase o trayectoria. Bandt y Pompe (BP) [?] introdujeron una metodología simbólica simple y robusta que toma en cuenta el ordenamiento temporal de las series temporales comparando valores vecinos en una serie temporal. La propiedad de causalidad de la PDF permite que los cuantificadores (basados en esta PDF) discriminan entre sistemas determinísticos y estocásticos [?]. Los datos simbólicos son: *(i)* creados por la clasificación de los valores de la serie; y *(ii)* definidos por el reordenamiento de los datos embebidos en orden ascendente, lo que equivale a una reconstrucción de espacio de fase con dimensión de embedding (longitud de patrón) D y retardo de tiempo τ . De esta forma, es posible cuantificar la diversidad de los símbolos de ordenación (patrones) derivados de una serie temporal escalar. Obsérvese que la secuencia de símbolos apropiada surge naturalmente de la serie temporal, y no se necesitan suposiciones basadas en modelos. El procedimiento es el siguiente:

- Dada una serie $\{x_t; t = 0, \Delta t, \dots, N\Delta t\}$, se genera una secuencia de vectores de longitud D .

$$(s) \longmapsto (x_{t-(d-1)\Delta t}, x_{t-(d-2)\Delta t}, \dots, x_{t-\Delta t}, x_t) \quad (2.13)$$

Cada vector resulta ser la "historia" del valor x_t . Evidentemente, cuanto más larga sea la longitud de los vectores D , mayor será la información sobre la historia de los vectores, pero se requiere un valor más alto de N para tener una estadística adecuada.

- Las permutaciones $\pi = (r_0, r_1, \dots, r_{D-1})$ de $(0, 1, \dots, D-1)$ es llamado "patrón de orden" de tiempo t , definido por:

$$x_{t-r_{D-1}\Delta t} \leq x_{t-r_{D-2}\Delta t} \leq \dots \leq x_{t-r_1\Delta t} \leq x_{t-r_0\Delta t} \quad (2.14)$$

Para obtener un resultado único se considera $r_i < r_{i-1}$ si $x_{t-r_i\Delta t} = x_{t-r_{i-1}\Delta t}$. De esta forma, todas las $D!$ permutaciones posibles π de orden

D , y la PDF $P = \{p(\pi)\}$ es definida como:

$$p(\pi) = \frac{\#\{s | s \leq N - D + 1; (s) \text{ has type } \pi\}}{N - D + 1} \quad (2.15)$$

En estas últimas expresiones, el símbolo $\#$ denota cardinalidad.

Por lo tanto, una distribución de probabilidad de patrones de orden $P = \{p(\pi_i), i = 1, \dots, D!\}$ se obtiene de la serie temporal. De esta manera, el vector definido por la ecuación (2.15) se convierte en un símbolo único π . Se establece $r_i < r_{i-1}$ si $x_{s-r_i} = x_{s-r_{i-1}}$ para la obtener una única solución. La única condición para la aplicabilidad del método BP es una suposición estacionaria muy débil: para $k \leq D$, la probabilidad para $x_t < x_{t+k}$ no debe depender de t . Con respecto a la selección de los parámetros, Bandt y Pompe sugirieron trabajar con $3 \leq D \leq 6$ para longitudes de series de tiempo típicas, y específicamente se consideró un retraso de tiempo $\tau = 1$ en su publicación principal.

Para destacar la diferencia entre una P causal y una no causal, consideremos una serie de valores $X = \{x_i, i = 1, 2, \dots\}$ generada por la función *randn* de Matlab's [©]; consideremos también la serie $Y = \{y_i, i = 1, 2, \dots\}$ como la resultante de ordenar la serie X en forma ascendente. Esto se puede ver en el ejemplo en la figura 2.12, en la figura 2.12a se muestran 1000 valores sorteados con una distribución uniforme entre 0 y 1, también mostramos en la figura 2.12b la versión ordenada de la serie de la figura 2.12a, son los mismos valores pero ordenados en forma ascendente. Una P no causal es el histograma normalizado que mostramos en las figuras 2.12c y 2.12d, en donde puede verse que $P(X)$ es idéntica a $P(Y)$, por lo que todos los cuantificadores que se calculen a partir de ellas serán idénticos para las dos series. Una P causal puede ser obtenida mediante el procedimiento de Bandt & Pompe descripto arriba, en este caso $P(X)$ de la figura 2.12e es bastante uniforme y $P(Y)$ de la figura 2.12f tiene una forma tipo delta. En este caso, P registra que Y es monótonamente creciente y presenta un solo patrón de orden.

Recientemente, la entropía de permutación se amplió para incorporar también información de amplitud. Ponderar las probabilidades de patrones individuales de acuerdo a su varianza mitiga los problemas potenciales con respecto a los patrones de "alto ruido, baja señal", porque los patrones de baja varianza que

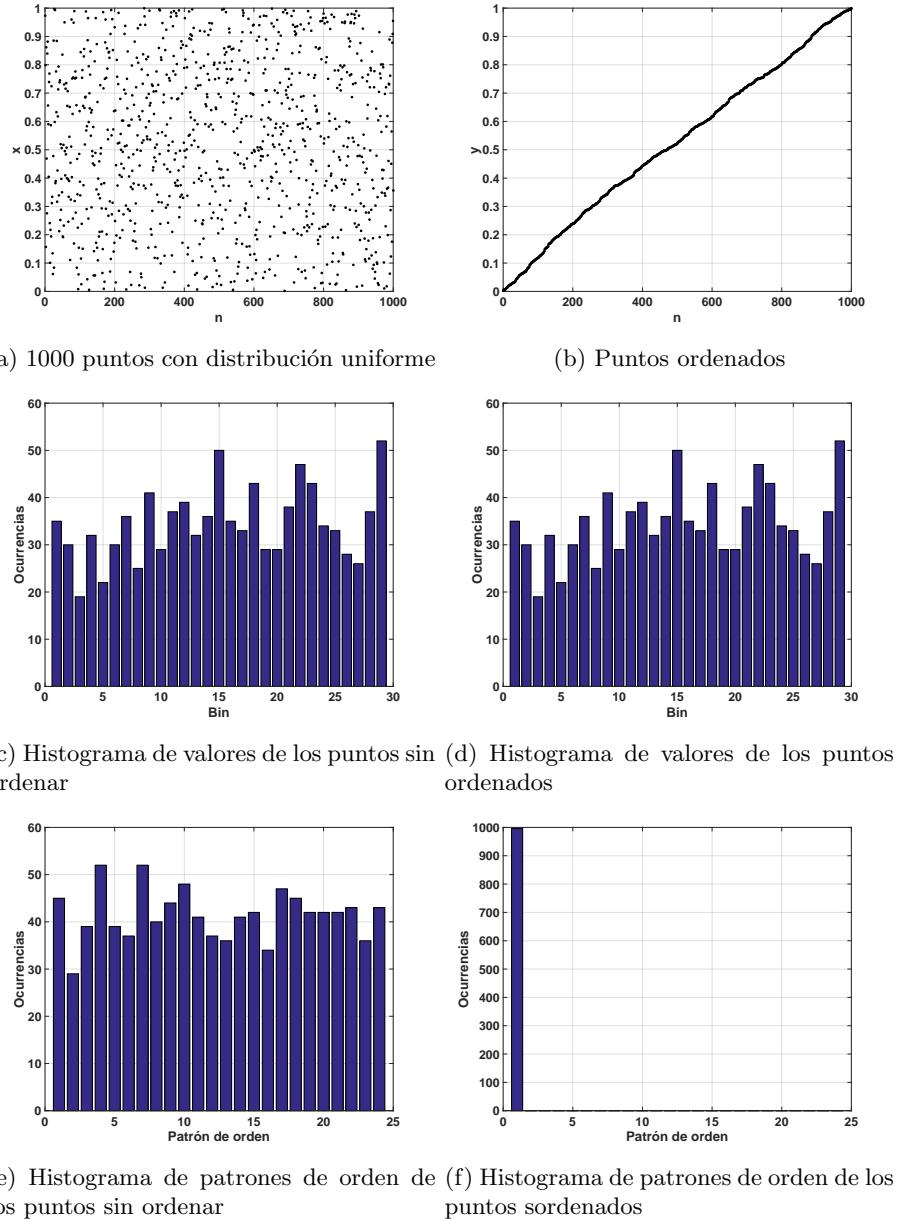


Figura 2.12: Comparación entre histogramas causal y no causal

están fuertemente afectados por el ruido se ponderan en las distribuciones de patrones ordinales ponderados resultantes. Por lo tanto, una posible desventaja de las estadísticas de los patrones ordinales, es decir, la pérdida de información de amplitud, se puede abordar mediante la introducción de pesos con el fin de obtener una “entropía de permutación ponderada (WPE)” [?]. Los pesos no normalizados se calculan para cada ventana temporal para la serie de tiempo X , tal que

$$w_j = \frac{1}{D} \sum_{k=1}^D \left(x_{j+k-1} - \bar{X}_j^D \right)^2. \quad (2.16)$$

En la ecuación anterior $x_{j+k-1} - \bar{X}_j^D$ denota la media aritmética del actual vector de embedding de longitud D y su varianza w_j se utiliza entonces para ponderar las frecuencias relativas de cada patrón ordinal p_j . Originalmente, se propuso esta técnica para discriminar patrones sumergidos en un bajo nivel de ruido. Nosotros también aprovechamos el hecho de que los puntos fijos no se computan en el WPE.

Se calculó la entropía de Shannon normalizada H y la complejidad estadística C de estas PDFs, y los valores obtenidos se denotan como:

- H_{hist} , es la entropía de Shannon normalizada aplicada a una PDF no causal P_{hist}
- H_{BP} , es la entropía de Shannon normalizada aplicada a una PDF causal P_{BP}
- H_{BPW} , es la entropía de Shannon normalizada aplicada a una PDF causal con contribuciones de amplitud P_{BPW}
- C_{BP} , es la complejidad estadística normalizada aplicada a una PDF causal P_{BP}
- C_{BPW} , es la complejidad estadística normalizada aplicada a una PDF causal con contribuciones de amplitud P_{BPW}

2.2.3. Planos de doble entropía y entropía-complejidad

Una visualización particularmente útil de los cuantificadores de la Teoría de la Información es su yuxtaposición en los gráficos bidimensionales. Se definen

cuatro planos de información:

1. Entropía causal vs. entropía no-causal, $H_{BP} \times H_{hist}$
2. Entropía causal con contribución de amplitudes vs. entropía no-causal, $H_{BPW} \times H_{hist}$
3. Complejidad causal vs. entropía causal, $C_{BP} \times H_{BP}$
4. Complejidad causal con contribución de amplitudes vs. entropía causal con contribución de amplitudes, $C_{BPW} \times H_{BPW}$

Estas herramientas de diagnóstico demostraron ser particularmente eficientes para distinguir entre el caos determinista y la naturaleza estocástica de una serie de tiempo ya que los cuantificadores de permutación tienen comportamientos distintos para diferentes tipos de procesos.

En la Fig. ?? se muestran los planos $H_{BP} \times H_{hist}$ y $H_{BPW} \times H_{hist}$ colapsados en un mismo plano. En este plano un valor más alto en cualquiera de las entropías, H_{BP} , H_{BPW} o H_{hist} , implica una mayor uniformidad de la PDF implicada. El punto $(1, 1)$ representa el caso ideal con histograma uniforme y distribución uniforme de los patrones de orden. Mostramos algunos puntos relevantes como ejemplo.

El ruido aleatorio blanco ideal con distribución uniforme da un punto en $(H_{hist}, H_{BP}) = (1, 1)$ representado por un círculo azul, un círculo rojo en la misma posición muestra los resultados cuando se incluyen las contribuciones de amplitud $(H_{hist}, H_{BPW}) = (1, 1)$. Si ordenamos el vector ideal con distribución uniforme de forma ascendente, los puntos resultantes se muestran con un cuadrado azul $(H_{hist}, H_{BP}) = (1, 0)$ y un cuadrado rojo $(H_{hist}, H_{BPW}) = (1, 0)$, este ejemplo ilustra la complementariedad de H_{hist} y H_{BP} .

Las estrellas azules y rojas muestran (H_{hist}, H_{BP}) y (H_{hist}, H_{BPW}) respectivamente aplicadas a una señal de diente de sierra. Los valores están perfectamente distribuidos en todos los intervalos, pero sólo aparecen unos pocos patrones de orden, esto explica el alto H_{hist} y bajo H_{BP} . La frecuencia de aparición de patrones de baja amplitud es mayor que los patrones de alta amplitud, entonces la PDF con contribuciones de amplitud es más uniforme y H_{BPW} es un poco más alto que H_{BP} . Cuando la señal de diente de sierra está contaminada con

ruido blanco, se incrementan H_{BP} y H_{BPW} como se muestra con triángulos azules y rojos. Es evidente que aparecen nuevos patrones de orden y tanto H_{BP} como H_{BPW} muestran valores más altos que los casos no contaminados, sin embargo el incremento de H_{BPW} es menor que H_{BP} mostrando que la técnica de registrar contribuciones de amplitud añade alguna inmunidad al ruido.

Finalmente, se evaluaron los cuantificadores de una secuencia de un mapa logístico que converge a un punto fijo, en todos los casos la longitud del vector de datos permanece constante y la longitud de transitorio es variable. Los resultados obtenidos sin las contribuciones de amplitud se representan en puntos azules, convergen a $(H_{hist}, H_{BP}) = (0, 0)$ a medida que la longitud de transitorio se hace más corta, sin embargo H_{BPW} (puntos rojos) permanece constante para todos los casos. El último punto en $(H_{hist}, H_{BP}) = (0, 0)$ corresponde a un vector de ceros, en este caso el histograma de patrones de orden con contribuciones de amplitud es también un vector nulo y H_{BPW} no se puede calcular. A través de este último ejemplo, mostramos que la convergencia a un punto fijo puede ser detectada por la información conjunta de H_{BP} y H_{BPW} .

En la figura ?? se muestra el plano causal $H_{BP} \times C_{BP}$. Podemos ver que no toda la región $0 < H_{BP} < 1, 0 < C_{BP} < 1$ es alcanzable, de hecho, para cualquier PDF los pares (H, C) de valores posibles caen entre dos curvas extremas en el plano $H_{BP} \times C_{BP}$ cite Anteneodo1996. Los mapas caóticos tienen entropía intermedia H_{BP} , mientras que su complejidad C_{BP} alcanza valores mayores, muy cercanos a los del límite de complejidad superior [?, ?]. Para procesos regulares, la entropía y la complejidad tienen valores pequeños, cercanos a cero. Los procesos estocásticos no correlacionados se ubican en la localización planar asociada con H_{BP} cerca de uno y C_{BP} cerca de cero. Los sistemas aleatorios ideales que tienen un Bandt & Pompe PDF uniforme, están representados por el punto $(1, 0)$ citeGonzalez2005 y una PDF tipo delta corresponde al punto $(0, 0)$.

En la figura ?? mostramos $H_{BP} \times C_{BP}$ con y sin contribuciones de amplitud. Se muestran los mismos puntos de muestra para ilustrar las posiciones planas para diferentes vectores de datos.

En ambos planos de información $H_{BP} \times H_{hist}$ en la Fig. 2.13 y $H_{BP} \times C_{BP}$ en Fig. 2.14, los datos estocásticos, caóticos y deterministas están claramente

localizados en diferentes posiciones planares.

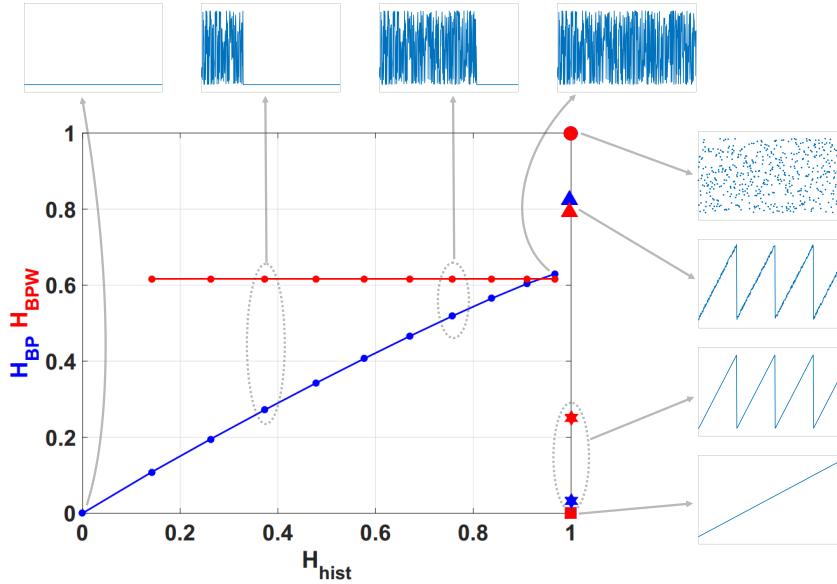


Figura 2.13: Causal-Non causal Entropy plane.

También usamos el número de patrones perdidos MP como un cuantificador [?]. Como mostraron recientemente Amigó y colaboradores [?, ?, ?, ?], en el caso de mapas deterministas, no todos los patrones de orden posibles pueden materializarse efectivamente en órbitas. De hecho, la existencia de estos patrones de orden faltantes se convierte en un hecho persistente que puede considerarse como una nueva propiedad dinámica. Por lo tanto, para una longitud de patrón fija (dimensión de embedding D) el número de patrones perdidos de una serie temporal (patrones no observados) es independiente de la longitud de la serie N . Obsérvese que esta independencia no caracteriza otras propiedades de la serie como la proximidad y la correlación [?, ?].

Existen fuentes bibliográficas para una discusión completa sobre la conveniencia de usar estos cuantificadores [?, ?, ?, ?, ?, ?, ?, ?].

Entropías diferenciales

La entropía de Shannon $S(P)$ es el punto de partida para otros cuantificadores: Para medir la entropía de una serie binaria es necesario !!!!!!!

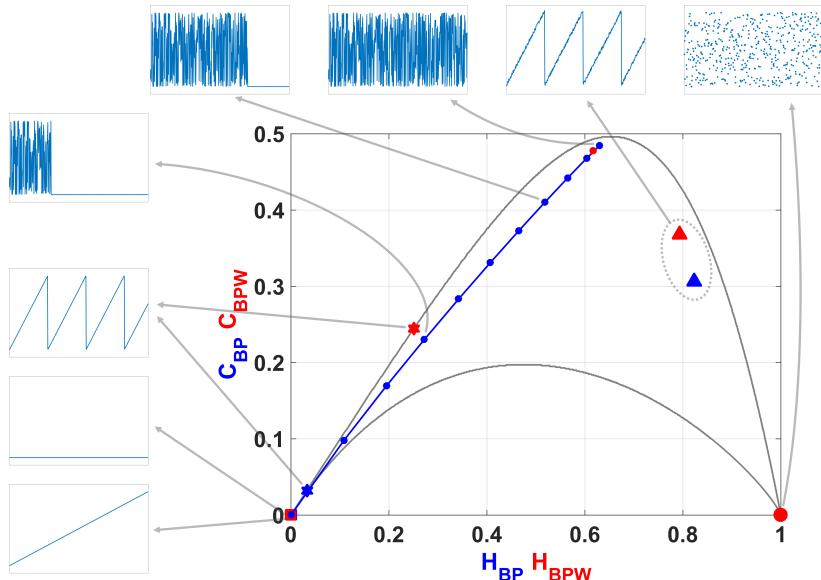


Figura 2.14: Causal Entropy-Complexity plane.

hablar del conjunto de particiones y no se que

1. Entropía normalizada $H(P)$: es la entropía de Shannon dividida por su valor máximo. Por ejemplo, si usamos S_2 (ver arriba), se obtiene la entropía máxima para equiprobabilidad entre dos símbolos. Su valor es $S_{max} = -1/2\log(1/2) - 1/2\log(1/2) = \log(2) = 1$; entonces, la entropía normalizada es $H_2 = S_2$. Si usamos S_W la equiprobabilidad entre las 2^W posibles palabras (números decimales de W -bits) produce $S_{max} = W$ y $H_W = S_W/W$. Finalmente, para $S_{BP}^{(D)}$ la equiprobabilidad entre los $D!$ patrones de orden produce $S_{max} = \log(D!)$ y $H_{BP}^{(D)} = S_{BP}^D/\log(D!)$.
2. Entropía diferencial o condicional h y h^* son:

$$h = S_{W+1} - S_W \quad (2.17)$$

$$h^* = S_{BP}^{(D+1)} - S_{BP}^{(D)} \quad (2.18)$$

En las expresiones de arriba $W = 1, 2, \dots$ y $D = 2, 3, \dots$, $S_0 = 0$ y $S_{BP}^{(1)} = 0$. Esta entropía diferencial o condicional da la cantidad promedio de información requerida para predecir el símbolo $(W + 1)$ (o $(D + 1)$), dado los

W (o D) símbolos precedentes.

3. Finalmente, las *rate entropies* h_0 y h_0^* [?, ?] son dadas por:

$$h_0 = \lim_{W \rightarrow \infty} h = \lim_{W \rightarrow \infty} S_W/W \quad (2.19)$$

$$h_0^* = \lim_{D \rightarrow \infty} h^* = \lim_{D \rightarrow \infty} S_{BP}^{(D)} / (D - 1) \quad (2.20)$$

Let us stress some important issues involved in the calculations of the above-mentioned entropies:

1. The binary entropy S_2 is noncasual while both, the block entropy S_W and the Bandt & Pompe entropy $S_{BP}^{(D)}$, are causal.
2. The block entropy S_W takes into account correlations between W consecutive bits. Bandt & Pompe entropy $S_{BP}^{(D)}$ takes into account correlations between D consecutive W -length words. Both grouping procedures (decimal numbers of W bits and permutation patterns of D decimal numbers) may be done with or without superposition. The number of data required for good statistics is different depending the grouping procedures are made with superposition or not.
3. For S_W there is only one grouping process (W bits are grouped to obtain a decimal numbers time series Y). Let us define α as a statistical quality parameter, given by the quotient between the number of elements in the symbolic time series Y and the number of symbols in the alphabet. In this paper we will not accept $\alpha < 10$.

Obviously the quality factor α increases with the length of the time series:

- a) If the grouping of W bits is made with superposition, two consecutive W -length words share $W - 2$ bits. Consequently starting with a file with a length of N -bits we get $N - W + 1$ words. Furthermore, there are 2^W symbols in the alphabet and $\alpha = (N - W + 1)/(2^W)$.
- b) If S_W is evaluated without superposition the number of W -length words is $\text{floor } \{N/W\}$ and the quality parameter becomes $\alpha = \text{floor } \{N/W\}/(2^W)$. For $N \gg W$ the statistical quality factor is W times lower than the one with superposition.

4. In the case of $S_{BP}^{(D)}$, there are two grouping processes involved.
- If both grouping processes are made with superposition we get $N - W - D + 2$ elements starting with a file N -bits length, and the quality factor is $\alpha = (N - W - D + 2)/D!$. In this case $S_{BP}^{(D)}$ takes into account the correlations between $W + D$ consecutive bits.
 - If the grouping process of W bits is made without superposition but the grouping of D decimal numbers is made with superposition we get $\lfloor N/W \rfloor - D + 1$ elements and the statistical quality parameter is $\alpha = (\lfloor N/W \rfloor - D + 1)/D!$. In this case $S_{BP}^{(D)}$ will include correlations between WD consecutive bits.
 - If the grouping process of W bits is made with superposition and the grouping of D decimal numbers is made without superposition we get $\lfloor (N - W + 1)/D \rfloor$ elements starting from a file with N bits. The statistical quality factor is $\alpha = \lfloor (N - W + 1)/D \rfloor/D!$ and $S_{BP}^{(D)}$ takes into account correlations between $W + D - 1$ bits.
 - If both grouping processes are made without superposition we get $\lfloor \lfloor N/W \rfloor /D \rfloor$ elements starting from a N -bits length file. The statistical quality factor is $\alpha = \lfloor \lfloor N/W \rfloor /D \rfloor/D!$ and $S_{BP}^{(D)}$ takes into account correlations between WD consecutive bits.

2.2.4. Cuantificador de entropías implementado en FPGA

En esta sección se describe la implementación de un sistema de medición de entropías. El diseño fue optimizado para ser implementado en un microcontrolador simple y pequeño, conservando una precisión aceptable. El sistema permite medir entropías a señales generadas internamente por código y a señales externas analógicas muestreadas. Se utilizó la placa de desarrollo *M1AFS-embedded kit* de ACTEL. En la FPGA (*Field Programmable Gate Array*) se instanció un microcontrolador 8051 al que se programó en lenguaje C. Se detalla el diseño del *hardware* y *software* y los resultados obtenidos.

Este trabajo se enmarca en un proyecto más ambicioso, que se propone el desarrollo e implementación en *hardware* de herramientas para el análisis de sistemas alineales. Contar con estas herramientas supondrá un avance significa-

tivo en el campo de la implementación de los sistemas no lineales. Permitiría comprender y describir con mayor precisión el comportamiento de la versión digital de este tipo de sistemas. El paquete completo de herramientas que nos proponemos implementar consta de:

- funcionales de la distribución de probabilidad: entropía de Shannon, desequilibrio estadístico y complejidad estadística;
- cuantificadores de la serie temporal, en especial exponentes de Lyapunov, autocorrelación, correlación cruzada y dimensiones fractales;
- operador de Perron Frobenius, y cuantificadores de diagramas de recurrencias;
- tests estadísticos propuestos en los bancos estandarizados para el estudio de generadores de números aleatorios (Marsaglia, NIST, etc.).

Al momento hay muy poca bibliografía sobre implementaciones en *hardware* de estas herramientas[?].

En el caso particular de la entropía es empleada en diversas aplicaciones, como por ejemplo, en la detección de anomalías en flujos de datos IP [?, ?]. En [?] se presentó un diseño y simulación en FPGA de un cuantificador de entropía, sin embargo actualmente no hay disponibles implementaciones en *hardware* de este cuantificador.

Dentro del proyecto mencionado, en este trabajo se implementa un sistema que calcula la entropía para la distribución de probabilidades (*PDF*) asociada a una serie de datos. Se analizan *PDF*'s causales y no causales. Los datos pueden tener un origen digital (generados mediante códigos), o bien provenir del muestreo de señales analógicas. Se utilizó la placa de desarrollo *M1AFS-embedded kit*, basado en el chip *M1AFS1500* que se destaca por tener un bloque analógico embebido en el mismo encapsulado de la FPGA. Luego, se verifica la exactitud numérica del cuantificador implementado comparando sus resultados con un programa patrón. A partir del máximo error detectado se determina la exactitud numérica del sistema.

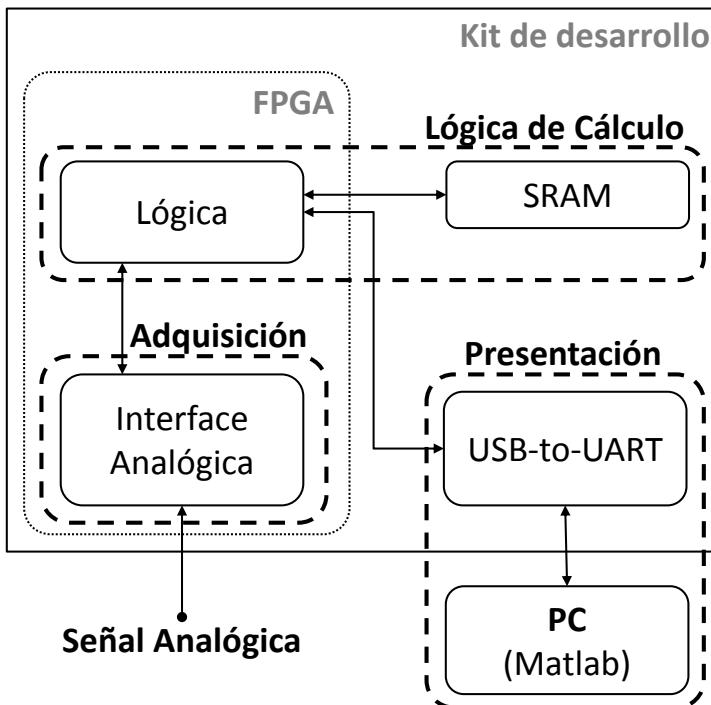


Figura 2.15: Esquema del sistema completo.

Hardware Implementado

El diseño del *hardware* se basó en el que provee ACTEL en [?], basado en el microcontrolador 8051, interfaces y periféricos. Fue realizado con el paquete de programas *Libero Soc v11.3[®]* de ACTEL. Se utilizó la placa de desarrollo *M1AFS-EMBEDDED-KIT* que contiene una FPGA *M1AFS1500* de ACTEL y periféricos [?]. El chip *M1AFS1500* contiene embebido un bloque analógico que consiste en nueve adaptadores direccionables de cuatro entradas cada uno, un multiplexor analógico de 32 entradas y un conversor analógico-digital configurable.

El sistema implementado puede dividirse en tres etapas principales como se muestra en la Fig. 2.15: una primer etapa de Adquisición de datos, que convierte a palabras digitales las señales del mundo analógico, una Lógica de Cálculo, que se vale de la memoria SRAM para llevar a cabo los cálculos y coordinar las interfaces y una etapa de Presentación de resultados, que envía los resultados de la medición a una computadora a través de la interfaz *USB-to-UART*.

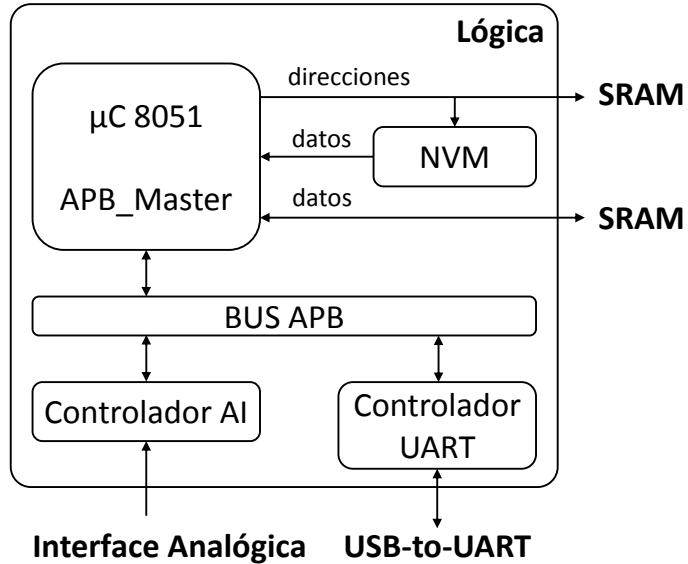


Figura 2.16: Detalle de la lógica de cálculo.

1. Etapa de Adquisición:

Para ingresar los datos analógicos a ser evaluados utilizamos la entrada de tensión *AV2* del *Analog Quad 2* del bloque analógico. Se encuentra direccionada en el canal siete del multiplexor analógico y fue configurada para un rango de tensiones de entrada de 0 V a 4 V. El conversor analógico-digital se configuró con una resolución de 12 bits. En este primer prototipo la frecuencia de muestreo máxima alcanzada fue de 16 ks/s limitada por el retardo necesario en el procesamiento de la lógica.

2. Lógica de cálculo:

En esta etapa se realizan los cálculos y la sincronización entre periféricos.

En la Fig. 2.16 pueden verse los bloques principales que la componen.

El núcleo de la implementación es un *Core 8051* que provee ACTEL en su catálogo de librerías. Se trata de un microcontrolador que contiene la lógica principal del microprocesador 8051 de Intel, sin sus periféricos. Este micro tiene una arquitectura Von Newman con un bus de direcciones de 16 bits, lo que limita nuestro diseño a 64 KB de memoria de código y 64 KB de memoria de datos.

Sobre este microcontrolador corre el programa que realiza los cálculos pre-

sentados en la sección ???. Se encarga de, a partir de los datos de entrada, obtener las PDFs (*BP* e *hist*) y de realizar los cálculos para la obtención de las entropías, según la ec. ???. El *software* implementado se describe más detalladamente en la sección 2.2.4.

La memoria de código es una memoria no volátil (NVM) implementada con los bloques flash internos de la FPGA. Ocupa las direcciones desde 0x0000 hasta 0xFFFF y se escribe con el contenido de un archivo en formato hexadecimal durante la compilación.

Las funcionalidades del sistema son ampliadas mediante la conexión de periféricos a través de la interfaz APB.

Para realizar la comunicación con la PC utilizamos el *Controlador UART*. La salida de este bloque es dirigida hacia afuera de la FPGA y se conecta a un chip *USB-to-UART* que se encuentra soldado a la placa del kit de desarrollo.

El bloque analógico es controlado por el *Controlador AI*, que direcciona y sincroniza sus entradas.

3. Presentación:

La etapa de Presentación de los datos involucra al chip adaptador *USB-to-UART* que se encuentra en la placa de desarrollo y es manejado tanto por el programa que corre en la FPGA como por el *software* que corre sobre la PC. El chip adaptador *USB-to-UART* es el responsable de adaptar la entrada-salida UART de la lógica a una entrada-salida USB estándar mediante la cual es posible interactuar con la PC. Por otra parte el programa que corre en la PC se encarga de la interfaz con el usuario y es descripto en detalle en la siguiente sección.

Software Implementado

El funcionamiento del sistema se logra mediante la interacción de dos programas. Uno corriendo en la PC y otro en el microcontrolador implementado en la FPGA. Puede verse un diagrama de flujo de ambos programas y la interacción entre ellos en la Fig. 2.17.

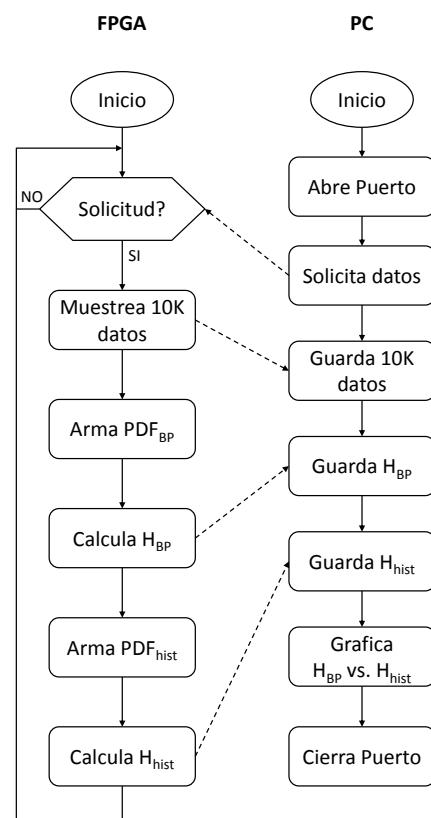


Figura 2.17: Diagrama de flujo del *software* implementado.

En la PC corre un *script* de *Matlab*[©] que se encarga de abrir el puerto serie en donde se encuentra mapeado el USB, solicitar los datos, tomar los resultados del mismo puerto, graficarlos en un plano H_{BP} vs. H_{hist} y cerrar el puerto.

Sobre el microcontrolador en la FPGA corre un programa escrito en lenguaje C y compilado para el microcontrolador 8051 utilizando la herramienta *SoftConsole IDE v3.4*[©]. El firmware es una modificación del usado en [?]. Cuando se presenta una solicitud de datos por el puerto UART, se guardan los datos muestrados de la entrada analógica. Luego, se recorre este vector generando las PDF_{hist} y PDF_{BP} , a las que se les calcula sus respectivas entropías H_{hist} y H_{BP} . Estos resultados son enviados a la PC mediante el mismo puerto.

Con el fin de validar el sistema, el programa en la FPGA envía a *Matlab*[©] el vector de datos muestrados, para que se puedan calcular en la PC sus entropías y compararlas con los resultados del sistema implementado.

Resultados

Como se dijo, para testear el sistema se compararon los resultados obtenidos por el sistema implementado y por un programa patrón que corre en la PC. Para esto, se generaron 10 000 muestras de señales con distintas formas de onda tanto externas (analógicas) como internas (digitales).

Las señales digitales fueron generadas por código en el microcontrolador, una corresponde a la función rand() de C y la otra al mapa caótico logístico con parámetro r=4.

Las señales analógicas fueron generadas con el generador de funciones *HP33120A*. Tienen una amplitud de 4 Vpp y un nivel de continua de 2 V de forma de aprovechar todo el rango del conversor analógico-digital y aumentar la relación señal-ruido. En los cuatro casos la frecuencia de las señales fue de 100 Hz y la velocidad de muestreo de 16 ks/s.

El cuadro 3.3 muestra el error absoluto entre los resultados de los cuantificadores calculados en la FPGA comparados con los resultados calculados con el programa patrón sobre los mismos datos.

La Fig. 2.18 muestra los valores entregados por la FPGA en el plano H_{BP} vs. H_{hist} .

Los resultados de la compilación nos permite conocer los recursos de la

Generador	Orígen	Error H_{BP}	Error H_{hist}
Rand	Digital	$1,7421E^{-6}$	$2,6977E^{-6}$
Logístico	Digital	$0,4256E^{-6}$	$94,693E^{-6}$
Triangular	Analógico	$6,3445E^{-6}$	$2,0028EE^{-6}$
Senoidal	Analógico	$6,3151E^{-6}$	$5,6506E^{-6}$
Cuadrada	Analógico	$0,1797E^{-6}$	$1,9930EE^{-6}$
Rampa	Analógico	$245,00E^{-6}$	$1,0876E^{-6}$

Cuadro 2.1: Error de los cuantificadores evaluados en la FPGA con respecto a los resultados calculados por el programa patrón.

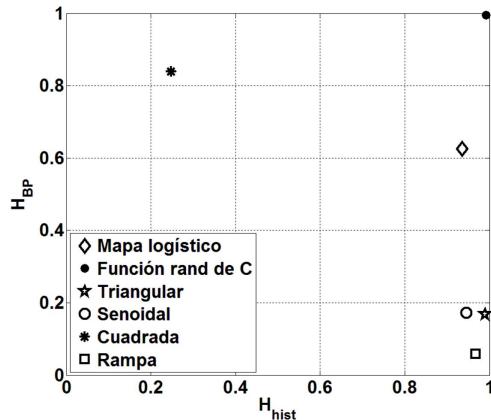


Figura 2.18: Resultados de las mediciones.

FPGA utilizados por el sistema completo y la cantidad de memoria ocupada por el *software* que corre en el microcontrolador. Recordemos que esta es una implementación de *hardware* rígida, es decir primero se arma el circuito en la FPGA (microcontrolador, periféricos, etc.) y luego se carga el *software* sobre él.

El reporte de la compilación de *hardware* devuelto el *Place and Route* se muestra en la Fig. 2.19. Podemos ver que la implementación utiliza un 19 % de los recursos lógicos de la FPGA, el 21 % de las celdas de entrada-salida y el 28 % de los bloques de memoria.

```

Core Cells      : 7349 of 38400 (19%)
IO Cells       : 53 of 252 (21%)

RAM/ROM Usage Summary
Block Rams : 17 of 60 (28%)
  
```

Figura 2.19: Recursos empleados por el *hardware* del sistema.

El reporte de la compilación de *software* se muestra en la Fig. 2.20. Podemos ver que la memoria FLASH no volátil se encuentra ocupada al 15,4 %. Por otro

lado, de las 65536 direcciones la memoria SRAM tenemos disponibles 61440 dado que parte de esta memoria es utilizada por el bus APB, por lo que se utiliza el 76,7 % de la memoria disponible.

Name	Start	End	Size	Max
PAGED EXT. RAM			0	256
EXTERNAL RAM	0x0000	0xb828	47145	65536
ROM/EPROM/FLASH	0x0000	0x276e	10095	65536

Figura 2.20: Recursos empleados por el *software* del sistema.

Discusión

El programa debió ser adaptado al microcontrolador instanciado en la FPGA. Estas modificaciones hacen que la salida del sistema implementado no sea igual a la de un programa que corre en la PC, al cual tomamos como programa o patrón. Por esto se testeó el error cometido, para tener una cota y determinar si los resultados de los cuantificadores son correctos. El programa patrón utiliza aritmética de 64 bits en punto flotante norma IEEE754-64 bits y emplea la librería math.h [?]. Para el algoritmo en la FPGA se disminuyó la aritmética a 32 bits de punto flotante norma IEEE754-32 bits. También se requirió el cálculo de la función logaritmo, que se implementó mediante un algoritmo de CORDIC. En el cuadro 3.3 se ve que el error absoluto no supera los $245E^{-6}$. Esto indica que se detecta diferencia recién a partir del quinto dígito decimal.

En la Fig. 2.18 puede verse como los cuantificadores H_{BP} y H_{hist} diferencian claramente las propiedades estadísticas de las series de datos analizadas. Las señales Senoidal, Rampa y Triangular presentan un valor alto de H_{hist} porque tienen casi todos los valores que es capaz de generar el conversor Analógico-Digital. Sin embargo, la mezcla de estos datos es mala por tratarse de una señales periódicas totalmente predecibles, esto se ve en el bajo valor de H_{BP} . Un caso interesante de analizar es la señal Cuadrada. El efecto del ruido aditivo es especialmente notable en las zonas en donde el valor de la señal debería ser constante. Se generan dos Gaussianas muy finas en torno a los valores ideales en la PDF_{hist} , esto no afecta demasiado el valor calculado H_{hist} , sin embargo para la PDF_{BP} , se calcula el patrón de orden directamente a la señal ruidosa, por lo que el valor de H_{BP} es más alto que el esperado. La señal generada mediante

la función rand de C, presenta las mejores propiedades estadísticas ubicándose en el punto $\sim (1, 1)$.

Conclusiones y trabajo futuro

Se desarrolló e implementó un sistema que permite medir con buena precisión las entropías causal y no-causal de señales analógicas provenientes del exterior de la FPGA y también internas generadas por código.

Se logró medir señales y realizar cálculos complejos con un microcontrolador modesto como el 8051 instanciado en la FPGA AFS1500 de ACTEL. Este primer prototipo cumple con las especificaciones de precisión y cantidad de recursos requeridos establecidas en el diseño, el próximo paso será optimizar el sistema en cuanto a frecuencia de operación e inmunidad al ruido.

Se prevé que el sistema permita modificar, en tiempo de ejecución, la frecuencia de muestreo, de forma de que sea adaptable a la señal de entrada, con el límite superior de 500 Ks/s fijado por el ADC.

Deberá agregarse también un umbral a partir del cual un valor es considerado distinto de otro, de esta forma se solucionaría el problema que presenta el ruido aditivo en el cálculo de H_{BP} .

El código de este sistema ocupa el 15,4 % del total de la memoria flash del micro instanciado, por lo que será posible agregar *software* para implementar otros cuantificadores y funcionalidades. En cuanto a los recursos disponibles en la FPGA se utilizaron 7349 celdas lógicas, quedando casi el 80 % de los recursos de *hardware* disponibles para implementar los sistemas bajo prueba en forma concurrente.

2.2.5. Dinámica de los ITQ's con AWGN y banda limitada

En esta sección exploramos la respuesta de un sistema de medición de entropías en presencia de ruido aditivo y señales filtradas. Esta inquietud surge como resultado de la implementación detallada en la sección 2.2.4. El filtrado es inherente al ancho de banda del sistema de medición y las señales a medir siempre están contaminadas con ruido, por lo tanto es necesario caracterizar la respuesta de nuestro sistema de medición ante estos dos procesos. Este trabajo es complementario al desarrollo de un sistema de medición de entropías

implementado en FPGA.

Filtrado digital

Ya sea en la elección de un filtro como en cualquier problema de diseño en ingeniería, generalmente no es posible dar una respuesta posible acerca de cual es al mejor solución. Se discute la posibilidad de la implementación de distintos filtros porque no hay un solo método de diseño ni un solo tipo de filtro mejor para todas las circunstancias. La elección del tipo de filtro depende de la importancia de sus ventajas aplicadas a cada problema.

Un filtro ideal es aquel en el que la respuesta en frecuencia es unitaria en el rango de las frecuencias de paso, cero en la banda de rechazo y no posee banda de transición. Dada la inherente periodicidad de la respuesta en frecuencia para tiempo discreto esta tiene la apariencia de un tren rectangular en frecuencias, sin embargo en este trabajo solo se muestra la frecuencia normalizada en el intervalo $(0; 1)$. Entonces la transferencia de un pasabajos ideal en frecuencia normalizada quedaría:

$$H_{LP} = \begin{cases} 1, & |f - 0,5| > f_c \\ 0, & |f - 0,5| < f_c \end{cases} \quad (2.21)$$

Ecuación definida en el intervalo de frecuencias normalizadas $f \in (0, 1)$.

El hecho de que no podemos contar con series de valores infinitamente largas para ser filtradas, equivale a decir que disponemos de una serie de muestras enventanada. Como el producto en el dominio del tiempo equivale a una convolución en el dominio de la frecuencia, podemos estudiar el efecto que este enventanado tiene sobre la respuesta frecuencial del filtro. Consideremos la ventana mas sencilla; la ventana rectangular. Supongamos que la aplicamos sobre una versión retardada de la respuesta ideal, su efecto en el dominio de la frecuencia será la convolución entre la respuesta de nuestro filtro ideal y la transformada esta ventana rectangular, es decir una función *sinc* de período $1/N$ en donde N es la cantidad de muestras que entran en la ventana.

El efecto de enventanado o truncamiento de la respuesta es doble: por una parte, la anchura del lóbulo principal está relacionada con la aparición de una banda de transición en el filtro. Por otra, la presencia de lóbulos laterales (se-

cundarios) lleva a la aparición de un ripple u oscilaciones en la respuesta en frecuencia, en ambas bandas, (más apreciable en la banda no pasante). La aparición de los lóbulos secundarios se debe a que la ventana rectangular presenta una discontinuidad abrupta que, al pasar al dominio de la frecuencia, conlleva un reparto de la energía por todo el espectro a causa del aliasing.

Una opción que se plantea es generalizar el concepto de ventana y emplear ventanas más suaves que la rectangular para realizar el truncamiento de la respuesta deseada, esta técnica es una de las formas de realizar un filtro FIR. Sin embargo, si analizamos la transformada de la ventana cuadrada vemos que presenta valores nulos cada $1/N$, que son los mismos lugares en donde aparecen las componentes espectrales de la DFT. Esto significa que los efectos de la ventana rectangular aparecen al convertir la respuesta de este filtro a tiempo continuo.

Otra opción sería diseñar un filtro analógico y transformar su respuesta a frecuencia discreta. Para esto contamos con fórmulas cerradas de diseño, por lo que podemos satisfacer cualquier especificación preestablecida. La utilización de esta técnica da como resultado un filtro IIR. Comparado con un FIR, un filtro IIR requiere un orden mucho menor para cumplir las especificaciones de diseño.

Aquí analizamos un sistema en el cual la respuesta es analizada en el dominio digital. Además, es necesario filtrar componentes espectrales de a una, lo que requiere una banda de transición muy estrecha, esto reduce el conjunto de filtros posibles. Por el lado del IIR probamos un filtro elíptico, este filtro presenta una banda de transición muy estrecha en sacrificio de un ripple que aparece tanto en la banda de paso como en la de rechazo. Por el lado del FIR probamos un filtro ideal con una ventana rectangular que abarca toda la serie de valores, en la sección 5.2 se detalla como fue implementado.

Resultados

Para representar la dinámica en función del filtrado, se eligieron dos señales representativas (cuadrada y senoidal) y se les calcularon los cuantificadores descriptos en la sección ?? luego de ser filtrados por los filtros elegidos en la sección 2.2.5. Por otro lado, se calculan los mismos cuantificadores a una señal de ruido blanco gaussiano.

En la figura 2.21 se muestra el procedimiento utilizado. Primero se generó un vector de ruido blanco gaussiano de $N = 50E3$ muestras, la desviación estándar σ es variable y se logra multiplicando al vector inicial de $\sigma = 1$ por la desviación estándar elegida. Luego se genera la señal determinística de $N = 50E3$ muestras, período $T = 100$ muestras y amplitud unitaria, que se sumó el ruido para lograr la señal contaminada. La señal resultante se filtra para luego calcular cuantificadores. Como se explicó más arriba, para calcular la entropía de valores se genera el histograma de valores y se lo normaliza para calcular la Función Densidad de Probabilidad de valores PDF_{hist} a la que se le calcula la entropía de Shannon normalizada que da como resultado la entropía de valores normalizada H_{hist} . Para calcular la entropía de patrones de orden se utiliza el histograma de patrones de orden que cuando se normaliza se consigue la función densidad de probabilidad de patrones de orden PDF_{BP} , a la que se le calcula la entropía de Shannon normalizada para conseguir la entropía de patrones de orden H_{BP} .

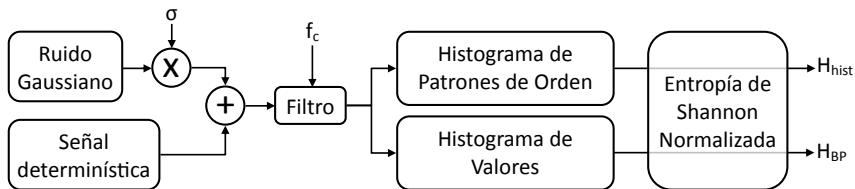


Figura 2.21: Diagrama de flujo del experimento.

Para evaluar la contribución de cada componente espectral a las entropías, se evaluaron dos filtros. Primero se aplicó un filtro elíptico de orden 10 con ripple pasabanda de $0,5dB$, ripple en la banda de rechazo de $100dB$ y frecuencia de corte variable f_c , en la figura 2.22 se muestra su respuesta en ganancia (fig. 2.22b) y fase (fig. 2.22c) para el caso de $f_c = 0,5$. De esta forma se logra un filtrado lo suficientemente abrupto como para considerar que a medida que se barren distintas frecuencias de corte se eliminan componentes espectrales individualmente. Los resultados de este filtrado se compararon con los resultados de un filtro ideal (fig. 2.23), que consiste en una máscara aplicada a la transformada de fourier de la señal a filtrar, de esta manera se consigue el espectro de la señal filtrada, el cual es antitransformado para recuperar la versión filtrada en las muestras. El diagrama de este filtro puede verse en la figura 2.23a. Este

procedimiento equivale a un filtrado ideal sin retardo, por lo que el bode de amplitud es $0dB$ en la banda de paso y $-\infty dB$ en la banda de rechazo (fig. 2.23a); la fase $\omega\tau = 0$ es lineal con pendiente nula (fig. 2.23c).

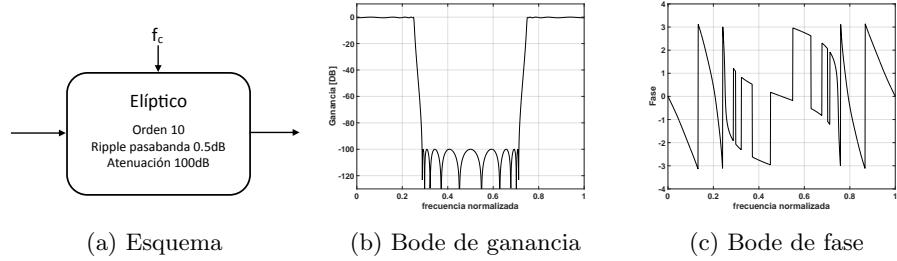


Figura 2.22: Filtro elíptico.

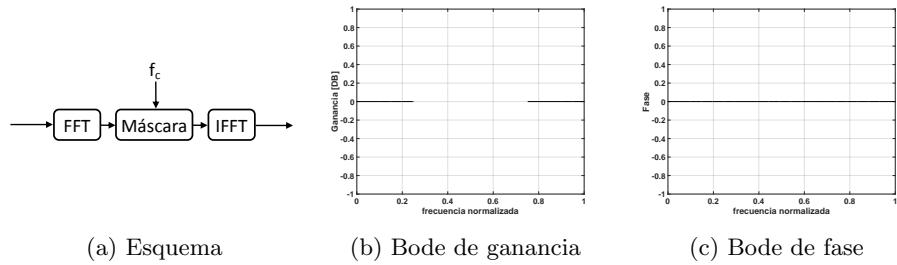
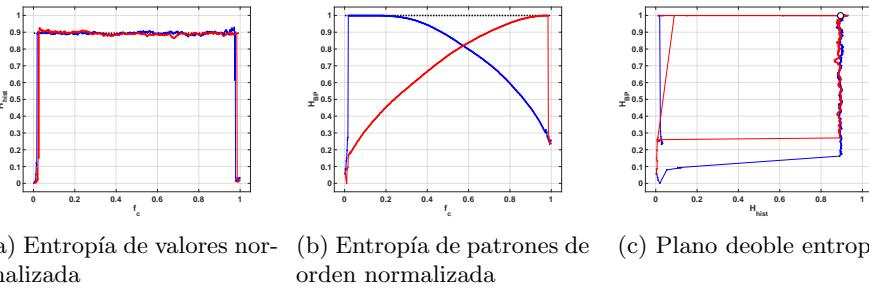


Figura 2.23: Filtro ideal.

Primero se aplicó una señal de ruido blanco gaussiano, es decir que la señal determinística es cero y la desviación estándar de la gaussiana unitaria. En la figura 2.24 se muestra el resultado de los cuantificadores a medida que se va barriendo la frecuencia de corte del filtro elíptico. En la figura 2.24a se muestra la entropía del histograma de valores H_{hist} , puede verse que su valor se mantiene constante alrededor de 0,9 tanto para el filtro pasa-bajos (roja) como el pasa-altos (azul), este valor es el mismo que resulta de calcular la entropía del histograma de valores a la señal sin filtrar (resultado que se muestra con una línea punteada negra en el mismo plot). También puede verse que cuando la frecuencia de corte del filtro elíptico se acerca a los extremos el valor del cuantificador cae, en estas frecuencias el método numérico que calcula el vector filtrado diverge debido a la precisión finita. En la figura 2.24b se muestra la entropía de los patrones de orden, H_{BP} se mantiene en valores bajos cuando el filtro (pasa-altos en azul y pasa-bajos en rojo) deja pasar pocas componentes

espectrales. Luego, a medida que la frecuencia de corte deja pasar más componentes espectrales, el cuantificador tiende a 1, que es justamente el valor que arroja cuando se ingresa con la señal sin filtrar (este valor se marca con una línea punteada negra). El cuantificador detecta los cambios en la forma de la señal a medida que es filtrada. Por último, en el plano $H_{hist} - H_{BP}$ de la figura 2.24c se compacta la información de ambos cuantificadores, aunque se pierde la noción de la frecuencia de corte.

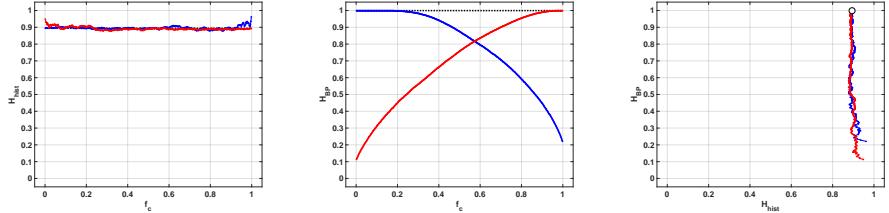


(a) Entropía de valores normalizados
 (b) Entropía de patrones de orden normalizada
 (c) Plano doble entropía

Figura 2.24: Cuantificadores calculados sobre la salida del filtro elíptico cuando se ingresa con ruido blanco gaussiano.

En la figura 2.25 se muestran los resultados del mismo procedimiento pero cuando se aplica un filtro ideal. El comportamiento de los cuantificadores es igual al del filtro elíptico en todos los casos con la diferencia que el método no diverge cuando $f_c \rightarrow 1$ o $f_c \rightarrow 0$. Pueden verse por lo tanto los valores que arrojan los cuantificadores en los extremos de la frecuencia de corte. La entropía no causal de la figura 2.25a aumenta levemente en los extremos, en donde el histograma de valores deja de tener una distribución gaussiana y se aplana levemente. También puede verse en 2.25b que la entropía de valores $H_{BP} \rightarrow 0,15$ cuando $f_c \rightarrow 0$ para el pasa-bajos (rojo) y para el pasa-altos (azul) $H_{BP} \rightarrow 0,22$ cuando $f_c \rightarrow 1$. En este caso es fácil comparar la sensibilidad al filtrado de ambos cuantificadores, en el plano doble entropía de la figura 2.25c. El círculo blanco muestra la posición en este plano cuando ningún filtro es aplicado, podemos ver que el apartamiento en el eje vertical aumenta a medida que la serie es filtrada, mientras que no se aparta en el sentido horizontal. Esto muestra que la sensibilidad al filtrado de H_{BP} es mucho mayor que la de H_{hist} .

Para el sistema planteado no se necesita volver al dominio continuo analógico, por lo que las dificultades mencionadas en la sección 2.2.5 respecto al filtrado

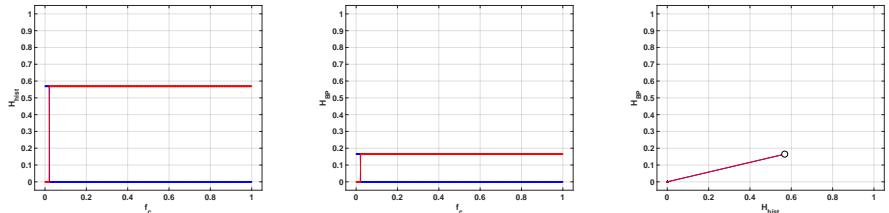


(a) Entropía de valores normalizada (b) Entropía de patrones de orden normalizada (c) Plano doble entropía

Figura 2.25: Cuantificadores calculados sobre la salida del filtro ideal cuando se ingresa con ruido blanco gaussiano.

ideal (como ripple en las bandas de paso y rechazo) no aplican a este caso. Por este motivo para esta serie de pruebas elegimos el filtro ideal, dado que presenta mejores resultados que el elíptico.

La primer señal determinística que se muestra es una senoidal de amplitud unitaria con período de 100 muestras, los resultados pueden verse en la figura 2.26. Mientras la única componente espectral no es filtrada, el valor de la entropía de valores es $H_{hist} \approx 0,57$ en la figura 2.26a y la entropía de patrones de orden $H_{BP} \approx 0,16$ en la figura 2.26b. Ambos cuantificadores caen a cero cuando la única componente espectral es filtrada, ya sea por el filtro pasa-bajos (azul) o por el pasa-altos (rojo). El plano doble entropía muestra un punto en $(0,57; 0,16)$ para la senoidal sin filtrar y otro en $(0; 0)$ cuando la única componente espectral es filtrada.

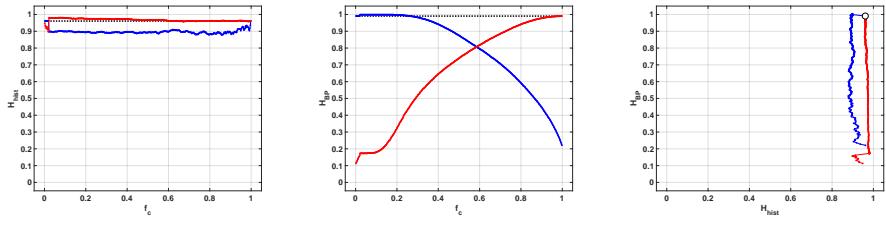


(a) Entropía de valores normalizada (b) Entropía de patrones de orden normalizada (c) Plano doble entropía

Figura 2.26: Cuantificadores calculados sobre la salida del filtro ideal cuando se ingresa con una senoidal limpia.

La salida de los cuantificadores cuando esta señal es contaminada con ruido gaussiano aditivo con $\sigma = 0,2$ puede verse en la figura 2.27. Vemos en la figura 2.27a que la entropía de valores aumenta cuando el filtrado no elimina la

componente espectral, dando valores incluso sobre el valor de la entropía de la gaussiana. Esto se debe a que la PDF de la senoidal es complementaria con la de la gaussiana, entonces la PDF de la resultante es más parecida a la del ruido uniforme. Para los patrones de orden de la figura 2.27b, el pasa-altos no deja ver un cambio significativo debido a que la componente espectral de la senoidal es eliminada en la zona en la que su entropía es alta. El pasa-bajos en cambio muestra que mientras esta componente está presente el valor de la entropía es asintótico a $H_{BP} \rightarrow 0,16$ a medida que la frecuencia de corte baja. Recordemos que $H_{BP} \approx 0,16$ es el valor de la entropía de patrones de orden de la senoidal limpia. En el plano doble entropía (figura 2.27c) se ve que ambos cuantificadores son complementarios, en el sentido que la entropía de valores detecta la presencia o no de la señal determinística mientras que la entropía de patrones de orden detecta el filtrado sobre la señal de ruido.

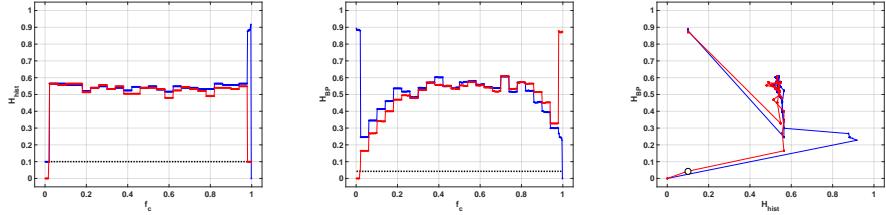


(a) Entropía de valores normalizada
 (b) Entropía de patrones de orden normalizada
 (c) Plano deble entropía

Figura 2.27: Cuantificadores calculados sobre la salida del filtro ideal cuando se ingresa con una senoidal ruidosa.

En la figura 2.28 se muestran los resultados cuando la señal determinística es una cuadrada sin ruido de amplitud unitaria y período de 100 muestras. Tanto la entropía de valores H_{hist} como la entropía de patrones de orden H_{BP} presentan una forma escalonada, sus valores se mantienen constantes a medida que se barre la frecuencia de corte de los filtros hasta que la siguiente componente espectral es filtrada. También se ve que en ambos casos los valores resultantes se mantienen bastante lejos del valor sin filtrar, que se muestra con una linea negra punteada.

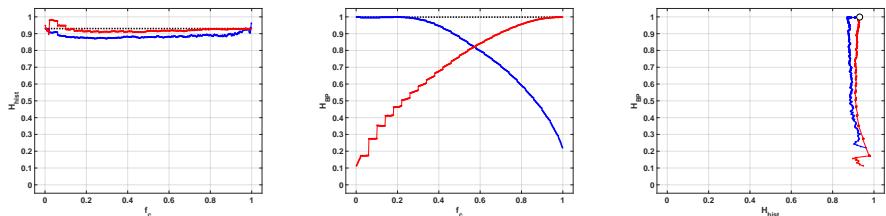
El caso contaminado con ruido (figura 2.29) cambia respecto del caso sin contaminar. En la figura 2.29a se ve que para el pasabajas (rojo) H_{hist} se mantiene alrededor del valor sin filtrar (linea punteada), excepto con las tres frecuencias



(a) Entropía de valores normalizada (b) Entropía de patrones de orden normalizada (c) Plano doble entropía

Figura 2.28: Cuantificadores calculados sobre la salida del filtro ideal cuando se ingresa con una cuadrada limpia.

más bajas, en donde su valor aumenta un levemente por las mismas razones que aumentaba con la senoidal contaminada. Algo parecido sucede con H_{BP} en la figura 2.29b. Cuando la cuadrada se contamina con ruido su valor se mantiene cercano al del ruido gaussiano, esto es por que en las regiones en las que la cuadrada es plana su contribución al patrón de orden es nula. Para el pasa bajos se ve un escalonado en la posición de cada componente espectral que se hace más notorio para las frecuencias más bajas, en donde la contribución del ruido ya es bastante baja y a la vez se encuentran las componentes espectrales de mayor peso. Esto no es tan notorio en el pasa-altos, en este caso cuando la contribución del ruido es de baja amplitud también lo es la de la determinística, enmascarando este fenómeno.



(a) Entropía de valores normalizada (b) Entropía de patrones de orden normalizada (c) Plano doble entropía

Figura 2.29: Cuantificadores calculados sobre la salida del filtro ideal cuando se ingresa con una cuadrada ruidosa.

Para caracterizar el comportamiento de los cuantificadores frente a la amplitud de ruido, se generaron cuadradas contaminadas con AWGN de dos amplitudes y se filtraron para calcular cuantificadores. En la figura 2.30 se muestran ambos cuantificadores cuando se hace variar el ruido con valores de la desvia-

ción estándar $\sigma = [0 \ 0,1 \ 1]$. Cuando comparamos las figuras 2.30a, 2.30b y 2.30c vemos un cambio significativo cuando pasamos de la señal limpia de 2.30a a la contaminada con bajos niveles de ruido de la 2.30b, sin embargo cuando pasamos del bajo nivel de ruido de 2.30b al de la figura 2.30c el cambio es mucho más sutil. De modo similar, entre las figuras 2.30d y 2.30e hay muy poco parecido, mientras que las figuras 2.30e y 2.30f son bastante similares. En este segundo caso es más evidente la diferencia cuando cambia el nivel de ruido, con bajos niveles puede verse el escalonado que aparece cada vez que una frecuencia es filtrada, mientras que cuando la amplitud de ruido es mayor este escalonado aparece solo en el pasabajo para las tres primeras frecuencias, que resultan ser las de mayor peso.

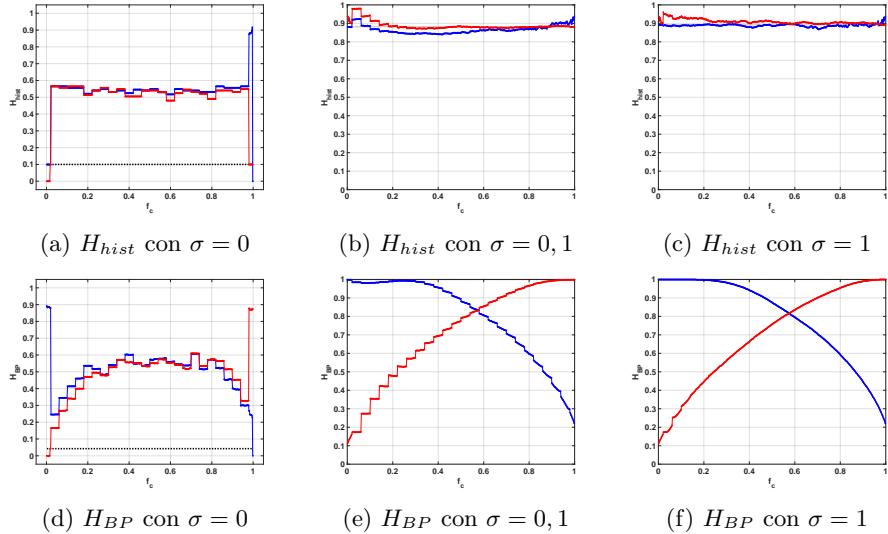


Figura 2.30: Cuantificadores calculados sobre la salida del filtro ideal cuando se ingresa con cuadradas contaminadas con AWGN de con amplitudes de ruido $\sigma = [0 \ 0,1 \ 1]$.

Discusión

Este trabajo fue necesario para explorar las fuentes de error en un medidor de entropías implementado en FPGA.

Para este primer análisis evaluamos que sucede aplicando un filtro abrupto, es por esto que elegimos para comparar un filtro elíptico y uno ideal. Las respuestas del filtro elíptico y del ideal fueron muy similares en el rango de

frecuencias en los que el elíptico tiene un buen comportamiento, sin embargo cuando la frecuencia de corte del elíptico se acerca a los extremos (es decir cuando $f_c \rightarrow 0$ o $f_c \rightarrow 1$) la salida del filtro diverge. El problema se debe a que el método numérico utilizado para calcular la salida del filtro diverge por la precisión finita utilizada. Como no necesitamos volver a la frecuencia continua nos quedamos con los resultados del ideal para hacer las pruebas, sin tener que preocuparnos por el ripple que aparece en las bandas de paso y rechazo cuando pasamos al mundo analógico.

Cuando comparamos las respuestas de los cuantificadores con y sin ruido, vemos que las señales limpias tienen mesetas, es decir que se mantienen constantes hasta que el filtrado elimina la siguiente componente espectral. Sin embargo, cuando son contaminadas con ruido los cuantificadores cambian para parecerse más a los resultados que arroja el ruido blanco gaussiano sin ninguna señal determinística. En todos los casos se vio que estos cuantificadores son muy sensibles a la presencia de ruido, los que nos permite vincular a este hecho los errores en la medición.

También vimos que los valores cambian a medida que se filtra la señal sin contaminar, lo que agrega una segunda fuente de error dada por el ancho de banda finito del sistema.

Para continuar con este proyecto faltaría, por un lado caracterizar el sistema de medición en cuanto a su ancho de banda y su rechazo al ruido aditivo, y por otro lado probar con otros cuantificadores (como complejidad, desequilibrio, entropía diferencial, rate entropy, etc) o con variantes de los presentados aquí (Bandt & Pompe pesada, amplitud promedio en el emmbedding, etc).

Capítulo 3

Generadores de Números Aleatorios Usando Caos

3.1. Sistemas Caóticos

Ha quedado claro que existen sistemas deterministas que rompen con el preconcepto de que los sistemas físicos pueden clasificarse en dos conjuntos disjuntos: sistemas deterministas y sistemas estocásticos. En esa concepción antigua un sistema determinista es aquél para el cual conocemos el modelo y por lo tanto es posible predecir con exactitud la evolución de sus variables de estado. Se utilizan en su descripción ecuaciones diferenciales o de recurrencia. Por otra parte un sistema estocástico es aquél para el cual el modelo no se conoce o se lo supone sumamente complejo como para ser obtenido, de modo que se adopta la estrategia de estudiar sus variables de estado en forma estadística. Se utilizan entonces en la descripción ecuaciones diferenciales o de recurrencia estocásticas.

El caos determinista demostró que complejidad en la evolución temporal no es sinónimo de complejidad en el modelo, cuando hay no linealidad: modelos deterministas muy simples originan señales de aspecto estocástico. La sensibilidad a las condiciones iniciales hace que en estos sistemas la predictibilidad sea a corto plazo (luego de un tiempo finito es imposible predecir la evolución) lo que ubica a estos sistemas en una posición intermedia entre determinista y

50 CAPÍTULO 3. GENERADORES DE NÚMEROS ALEATORIOS USANDO CAOS

estocástico.

Como consecuencia se desarrollaron en los últimos años un número creciente de aplicaciones de los sistemas caóticos, empleándolos principalmente como generadores de ruido controlado, generadores de números pseudoaleatorios, portadoras de señales, sistemas de encriptado, etc.

Hoy en día, los sistemas dinámicos son un objeto de estudio interdisciplinario, aunque originalmente fue una rama de la física. Todo comenzó a mediados del 1600, cuando Newton inventó las ecuaciones diferenciales, descubriendo sus leyes del movimiento de gravitación universal, y las combinó con las leyes de Kepler sobre el movimiento planetario. Específicamente, Newton resolvió el problema de los dos cuerpos (por ejemplo el sistema tierra-sol).

Subsecuentes generaciones de matemáticos y físicos intentaron extender los métodos analíticos de Newton al problema de los tres cuerpos (por ejemplo luna-tierra-sol), pero curiosamente para resolver este problema se necesitó mucho más esfuerzo. Luego de décadas de esfuerzo, se dieron cuenta de que el problema de los tres cuerpos era esencialmente imposible de resolver, en el sentido de obtener las fórmulas explícitas.

La ruptura vino con el trabajo de Poincaré a finales del 1800. Él introdujo un nuevo punto de vista que enfatizaba las cuestiones cualitativas más que las cuantitativas (por ejemplo, ¿es estable el sistema luna-tierra-sol?). Poincaré desarrolló una poderosa aproximación geométrica que es usada hoy para estudiar sistemas dinámicos y también fue el primero en vislumbrar la posibilidad de caos, en el cual un sistema determinístico exhibe un comportamiento aperiódico que depende sensiblemente de las condiciones iniciales, haciendo así imposible la predicción a largo plazo.

Pero el caos se mantuvo en segundo plano hasta la segunda mitad del 1900, en donde los osciladores no lineales jugaron un rol vital en el desarrollo de tecnologías de radio, radar, lazos de enganche de fase y láser. Por el lado matemático, los osciladores no lineales también estimularon la invención de nuevas técnicas matemáticas. Los métodos geométricos de Poincaré se fueron extendiendo para producir un conocimiento mucho más profundo de la mecánica clásica.

La invención de la computadora por el 1950 fue una línea divisoria en la historia de los sistemas dinámicos. La computadora nos permite experimentar

con ecuaciones en una forma que antes era imposible, y así desarrollar alguna intuición acerca de los sistemas no lineales. Estos experimentos llevaron a Lorenz a descubrir en 1963 el movimiento caótico de un atractor extraño, mientras estudiaba un modelo simplificado de la circulación de convección para comprender mejor la notoria impredictibilidad del clima. Lorenz encontró que la solución a sus ecuaciones nunca caían al equilibrio o a un estado periódico. Además, si comenzaba sus simulaciones de dos condiciones iniciales ligeramente diferentes, los comportamientos resultantes pronto serían totalmente diferentes. Como consecuencia de ello, el sistema es inherentemente impredecible, pequeños errores en las mediciones del estado actual de la atmósfera (o cualquier sistema caótico) sería amplificado rápidamente. Pero Lorenz también mostró que había estructura en el caos, cuando fueron pleteadas en tres dimensiones, las soluciones a sus ecuaciones cayeron sobre un set de puntos en forma de mariposa. Él sostuvo que este sistema tenía que ser “un infinito complejo de superficies”. Lo que hoy podríamos considerar como un ejemplo de fractal.

El trabajo de Lorenz tuvo un pequeño impacto hasta 1970, los años del boom del caos. Se desarrollaron teorías completamente nuevas basadas en consideraciones sobre atractores caóticos, como turbulencia de fluidos y biología de las poblaciones y se encontraron comportamientos caóticos en reacciones químicas, circuitos electrónicos, osciladores mecánicos, semiconductores y oscilaciones biológicas como el ritmo cardíaco y circadiano.

Hoy, la teoría del caos es un herramienta más para el estudio de sistemas dinámicos y los sistemas caóticos forman parte de una gran cantidad de dispositivos.

En algunas aplicaciones puede interesar, más que conocer las ecuaciones explícitas de las soluciones de un sistema, poder analizar sus propiedades cualitativas, tales como la periodicidad, el comportamiento cuando crece la variable independiente (la que se supone que es el tiempo), si es constante, o si se aproxima a una solución conocida, etc. Una herramienta útil en este sentido es el diagrama de fase. El espacio de fase es el lugar geométrico que ocupan las posibles soluciones del sistema de ecuaciones diferenciales, en él se dibujan las trayectorias que son solución a un sistema de ecuaciones. La teoría cualitativa intenta clasificar los sistemas en función del tipo de trayectorias que poseen, en

lugar de intentar resolver las ED's.

Sin embargo, el teorema de existencia y unicidad de las soluciones a un sistema de ecuaciones garantiza que si f es continuamente diferenciable, los campos vectoriales sobre el espacio de fases son suaves y cada punto de este espacio tiene solución única. La existencia de este teorema tiene un corolario importante: trayectorias diferentes nunca se intersectan. Como consecuencia de esto, las trayectorias sobre el plano de fases quedan restringidas a: un nodo estable o inestable, centro, foco estable o inestable y puerto. Queda claro, entonces que un sistema continuo con derivada continua debe tener dimensión mayor o igual a tres para que pueda ser caótico, además, la trayectoria en el espacio de fases debe ocupar un dominio restringido.

3.2. El problema de la Aritmética Discreta

3.3. Caos en redes neuronales

El problema de el caos en las redes neuronales ha recibido mucha atención recientemente. Las actividades en este campo pueden ser divididas en tres categorías:

- Intentos por explicar experimentalmente el comportamiento aperiódico observado en una sola neurona perturbada o en un pequeño ensamble de neuronas.
- Intentos por explicar el comportamiento temporal complejo del cerebro y los posibles roles del caos en el procesamiento de la información.
- El estudio de las rutas al caos y las propiedades de los atractores caóticos en en modelos de redes neuronales.

Las redes neuronales artificiales proveen soluciones efectivas a problemas en diversos campos, en particular, pueden servir como generadores de señales caóticas. Las aplicaciones de señales caóticas son muy diversas, pero en este caso son especialmente atractivas ya que en los algoritmos de aprendizaje se utiliza una búsqueda aleatoria, en estos casos un generador neuronal de caos puede ser una parte de la red neuronal determinística que se está entrenando.

3.3.1. El modelo de Hopfield

Una de las piedras fundamentales para el reciente renacimiento en el campo de las redes neuronales fué el modelo asociativo propuesto por Hopfield en 1982. La aproximación de Hopfield es un enfoque teórico para pensar ensambles entre unidades de cómputo.

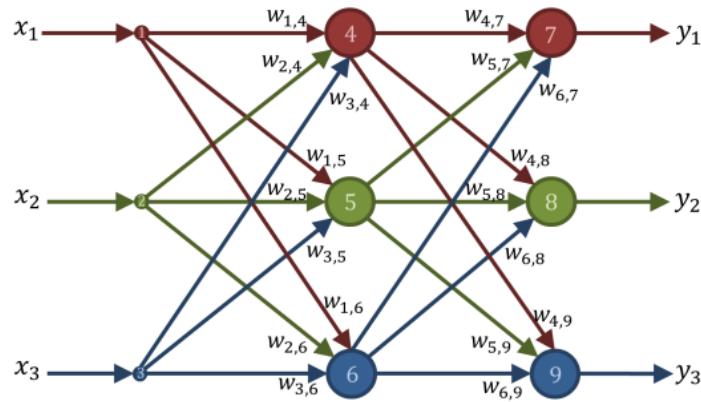
El perceptrón multicapa es una RNA formada por capas de neuronas. Las neuronas pueden pertenecer a la capa de entrada, capas ocultas o capa de salida. Estas neuronas no incorporan memoria por lo que su salida depende del estado de sus entradas en el instante actual (no tienen retardo), además, como el nombre de sus capas lo sugiere, las conexiones son hacia adelante. Es por esto que la matriz de pesos tiene solo algunos valores distintos de cero, no hay conexiones hacia atrás, ni en la misma capa, ni sobre la misma neurona, ni saltándose capas. En la figura 3.1 se ve un ejemplo para un perceptrón pequeño y su matriz de pesos.

Al contrario de los preceptrones multicapa ,los sistemas adaptativos y los mapas autoorganizados, las redes de Hopfield si tienen realimentación entre neuronas. Este tipo de arquitectura tiene como campo principal de aplicación la optimización de procesos. Se basa en el planteamiento de una memoria asociativa; se hace necesario entonces definir una una función de energía. Pone de manifiesto la analogía existente entre su modelo y la física estadística clásica, lo que permite usar sus bien conocidas herramientas matemáticas. Además, es muy interesante que se destaca la facilidad de implementación en FPGA y VLSI.

Esta red recurrente se basa en almacenar información en un sistema que presenta una configuración dinámica estable, es decir, se plantea como una memoria asociativa o memoria direccionable por contenido. Intuitivamente, la idea de Hopfield es localizar cada patrón que se requiere almacenar a la red en el fondo de un valle de la función de energía. Se parte de un determinado estado inicial (información de partida) tras lo cual se deja evolucionar el sistema hasta llegar a un estado estable. Este estado estable será el patrón que se corresponde con nuestro estado inicial (reconocimiento de patrones).

Hopfield, en su trabajo destaca tres diferencias con el perceptrón multicapa:

- Su modelo incluye realimentaciones, que son básicas en su modo de funcionamiento.



$0 \ 0 \ 0$	$w_{1,4} \quad w_{1,5} \quad w_{1,6}$	$0 \ 0 \ 0$	Vienen de la capa 1
$0 \ 0 \ 0$	$w_{2,4} \quad w_{2,5} \quad w_{2,6}$	$0 \ 0 \ 0$	
$0 \ 0 \ 0$	$w_{3,4} \quad w_{3,5} \quad w_{3,6}$	$0 \ 0 \ 0$	
$0 \ 0 \ 0$	$0 \ 0 \ 0$	$w_{4,7} \quad w_{4,8} \quad w_{4,9}$	Vienen de la capa 2
$0 \ 0 \ 0$	$0 \ 0 \ 0$	$w_{5,7} \quad w_{5,8} \quad w_{5,9}$	
$0 \ 0 \ 0$	$0 \ 0 \ 0$	$w_{6,7} \quad w_{6,8} \quad w_{6,9}$	
$0 \ 0 \ 0$	$0 \ 0 \ 0$	$0 \ 0 \ 0$	Vienen de la capa 3
$0 \ 0 \ 0$	$0 \ 0 \ 0$	$0 \ 0 \ 0$	
$0 \ 0 \ 0$	$0 \ 0 \ 0$	$0 \ 0 \ 0$	
$0 \ 0 \ 0$	$0 \ 0 \ 0$	$0 \ 0 \ 0$	

Llegan a la capa 1 Llegan a la capa 2 Llegan a la capa 3

Figura 3.1: Perceptrón multicapa y matriz de pesos asociada. Puede verse que la topología de la red y la configuración de la matriz de pesos son biunívocas.

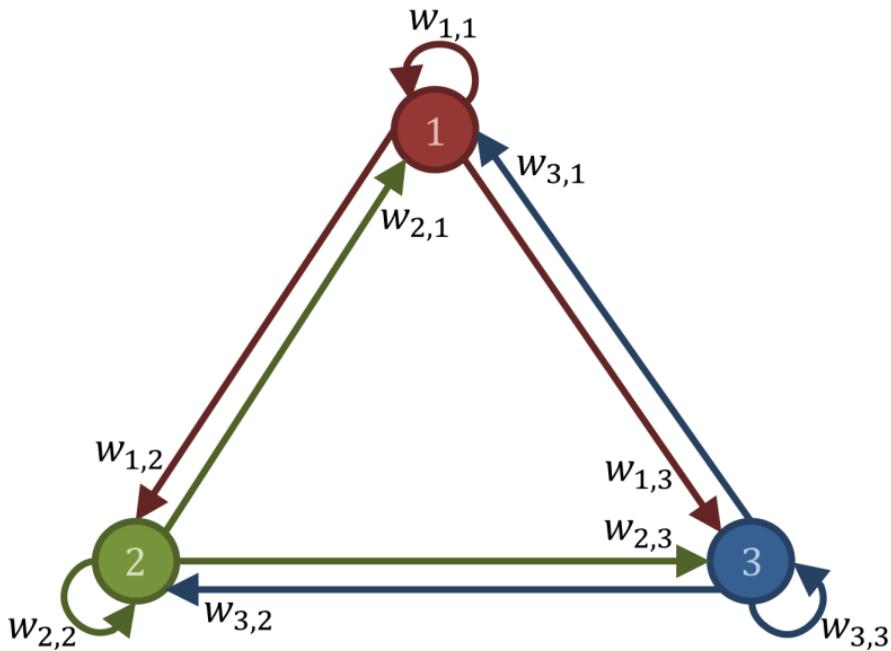


Figura 3.2: Red de Hopfield. Ahora, la matriz de pesos tiene todos sus valores permitidos.

- La elección de la arquitectura del perceptrón multicapa se realiza en forma arbitraria.
- El perceptrón multicapa funciona de manera síncrona, es decir, todas las neuronas cambian al mismo tiempo. La red de Hopfield permite un funcionamiento tanto síncrono como asíncrono, aunque el funcionamiento asíncrono es el más habitual en las neuronas biológicas.

El grafo de la red cambia con respecto al perceptrón multicapa, la representación no es la de un grafo separable por capas con conexiones hacia adelante, sino la de un grafo completo como se ve en la figura 3.2.

3.3.2. Un caso de estudio

La red neuronal usada tiene el modelo de tiempo continuo

$$\dot{u} = -u + W \cdot f(u); u \in \mathbb{R}^3 \quad (3.1)$$

en donde u es un vector de tres dimensiones, W es la matriz de pesos y f es la función de activación

$$u = \begin{pmatrix} x \\ y \\ z \end{pmatrix}; W = \begin{pmatrix} w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,1} & w_{2,2} & w_{2,3} \\ w_{3,1} & w_{3,2} & w_{3,3} \end{pmatrix}; f = \begin{pmatrix} \arctan x \\ \arctan y \\ \arctan z \end{pmatrix} \quad (3.2)$$

Ecuaciones que se corresponden con el diagrama de la figura 3.2. Vemos de la ecuación que se corresponde con una red de Hopfield de memoria diferencial. No disponemos de computadoras analógicas, por lo que el sistema debe ser convertido a tiempo discreto. Aunque el paquete de programas Matlab incluye rutinas para el cálculo de ecuaciones diferenciales, perdemos el control de paso de tiempo necesario para calcular el exponente de Lyapunov con el método descripto en la sección 2.1, además lo necesitamos para una futura implementación en hardware. Usamos para esto una aproximación de Euler de primer orden, en donde la derivada se aproxima con un trapecio de base Δt .

$$\begin{aligned} \frac{u_{n+1} - u_n}{\Delta t} \approx \dot{u}_n &= -u + W \cdot f(u_n) \Rightarrow & (3.3) \\ \Rightarrow u_{n+1} &= (1 - \Delta t)u_n + \Delta t W \cdot f(u_n) \\ &= Gu_n + \Omega f(u_n) \end{aligned}$$

En la figura 3.3 se muestra nuestra nueva red neuronal en tiempo discreto. Sus coeficientes dependen del paso de tiempo. Este sistema se aproxima al de tiempo continuo en el límite $\Delta t \rightarrow 0$, en nuestro caso se verificó que el sistema converge al planteado. Pudo verse que con $\Delta t = 1$ y $\Delta t = 0,1$ las soluciones en el espacio de fases fueron las mismas, para hacer los cálculos utilizamos $\Delta t = 0,01$.

Se barrió un parámetro (peso de un axón) para identificar la existencia de caos en función de éste. Siguiendo a [?] en donde se reporta una transición al caos en torno a un juego de parámetros, utilizamos la siguiente matriz de pesos: en donde p es el parámetro a barrer entre $-0,35$ y $0,55$ en pasos de 9×10^{-5} .

Para cada valor del parámetro se le da condiciones iniciales al sistema $[1, 68; -0, 292; -3, 47]$ y se lo deja evolucionar $800s$, esto es $200s$ más que el transitorio más largo reportado en [?], con esto nos aseguramos de descartar el transitorio y que el sistema

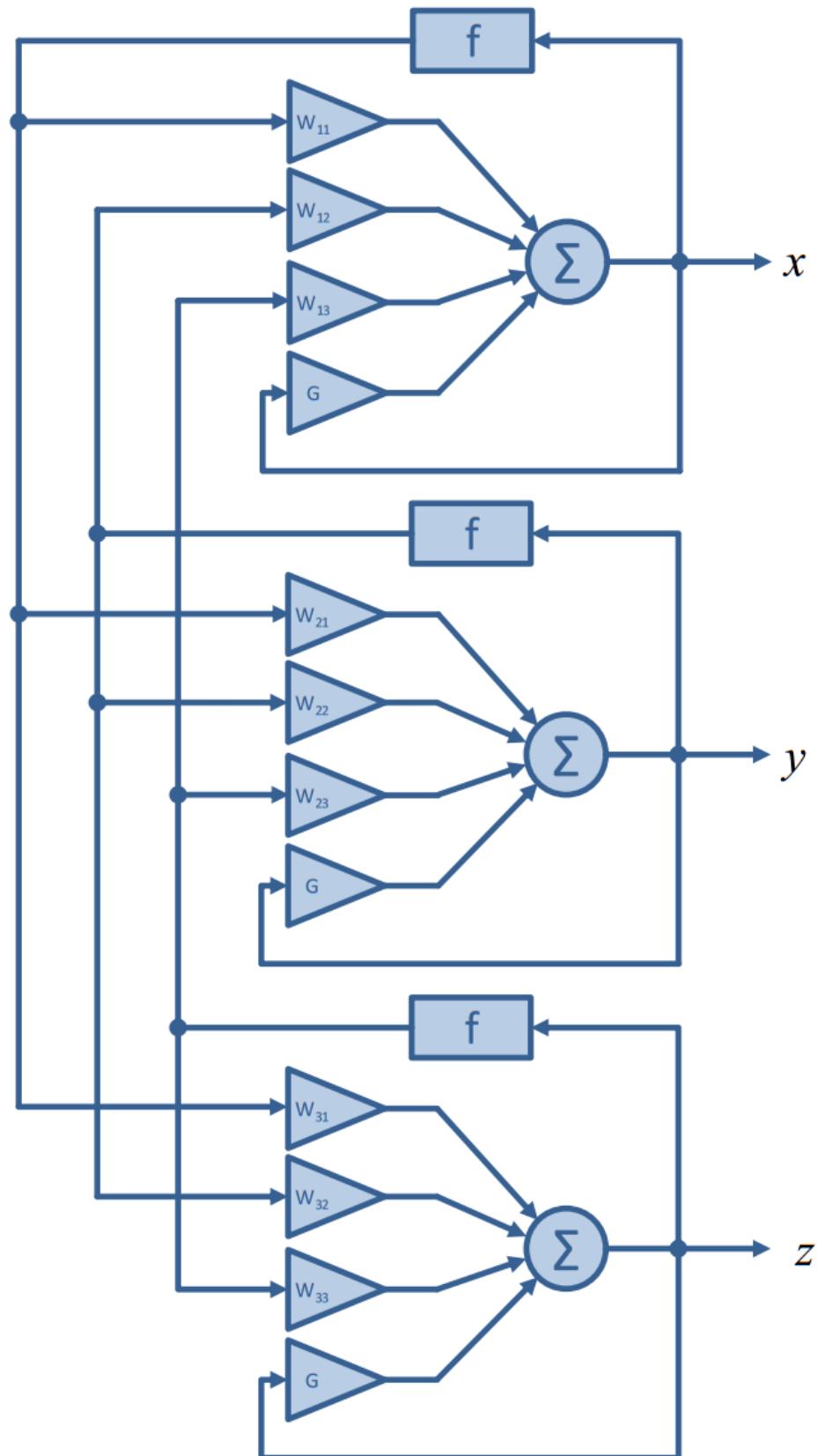


Figura 3.3: Red utilizada. Se trata de una red de Hopfield tridimensional de memoria diferencial, el diseño está orientado a una posterior implementación en FPGA.

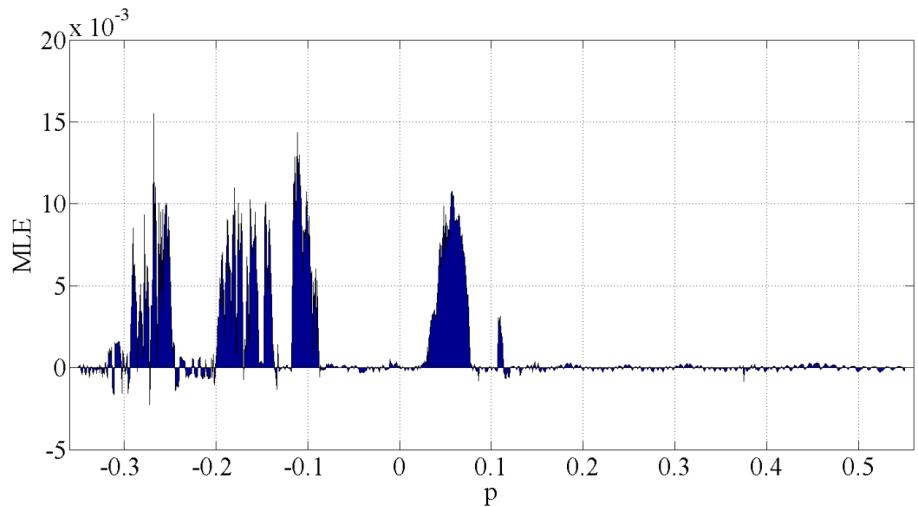


Figura 3.4: Exponente de Lyapunov en función del parámetro p . Existe caos en toda la zona en la que es positiva.

se encuentra en régimen permanente. Se calcula el MLE para $t \in (800; 1000]$.

De esta forma se genera la figura 3.4 en donde se muestra el MLE en función del parámetro. Como es usual, el MLE no es una función suave, sino que es una función discontinua que presenta saltos abruptos en todo el dominio, sin embargo, se encontraron zonas de caos robusto frente al parámetro p en algunos intervalos, especialmente en $p \in (0, 0223; 0, 0791)$, esto significa que el caos persiste con una variación no infinitesimal del parámetro, esta zona es muy buscada para implementaciones prácticas.

Para mostrar la transición al caos y la relación entre el MLE y el espacio de fases, se eligieron dos parámetros $p_1 = -0, 2725$ y $p_2 = 0, 268$, para p_1 el $MLE = -2, 2 \times 10^{-3}$, para p_2 el $MLE = 1, 55 \times 10^{-2}$. Se muestra la trayectoria resultante para cada uno en la figura 3.5.

Para el atractor caótico, dos trayectorias generadas a partir de condiciones iniciales muy cercanas deben, al cabo de un tiempo, separarse y oscilar en trayectorias distintas. En la figura 3.6 puede verse este efecto.

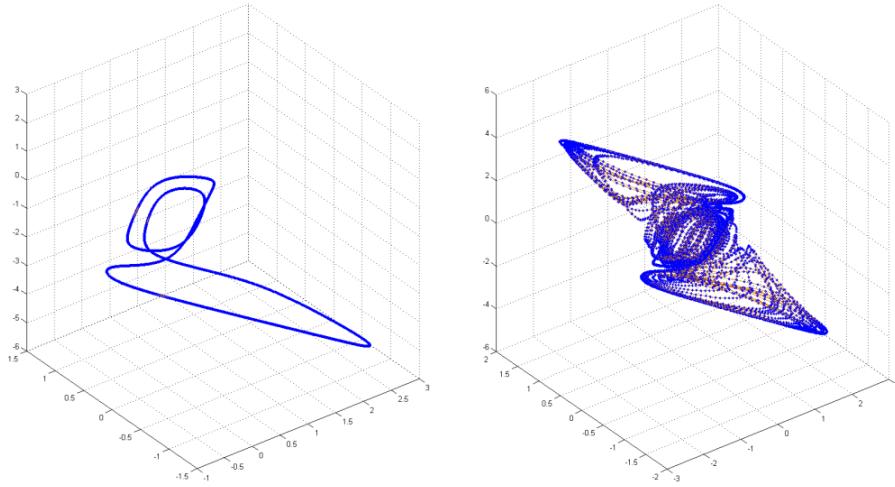


Figura 3.5: Dos trayectorias características del sistema en el espacio de fases. La trayectoria de la izquierda se corresponde con un $MLE < 0$ y la positiva con un $MLE > 0$.

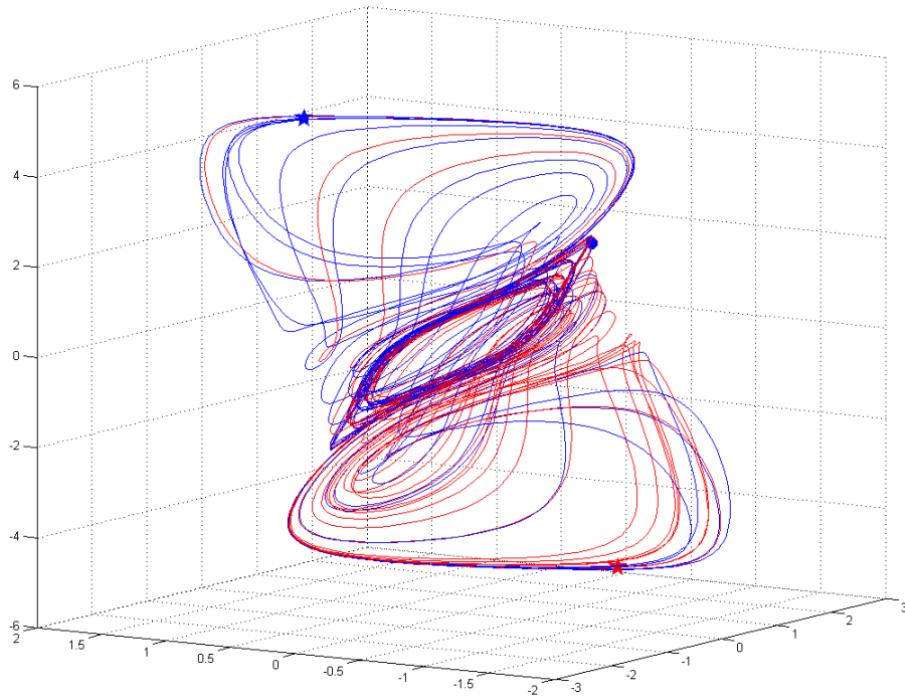


Figura 3.6: Dos trayectorias de la red de Hopfield para condiciones iniciales próximas. Las condiciones iniciales están marcadas con dos puntos grandes cerca del centro del atractor y los valores después de $\Delta t = 3s$ con estrellas en ambos extremos de la figura.

3.4. Cripto-codificación caótica variante en el tiempo

3.4.1. abstract

En este trabajo se presenta una nueva técnica para la criptocodificación de datos mediante una familia de mapas caóticos. El diseño se basa en un mapa cuadrático bidimensional que tiene la característica de modificar su atractor según los valores que tomen sus 12 coeficientes reales. Esto hace que este sistema sea muy atractivo al momento de su implementación en hardware. Para la implementación se utilizó aritmética de punto fijo con 19 bits. Se realizaron simulaciones y el diseño en VHDL mediante el programa Quartus II v8.0 de ALTERA, para su posterior implementación en FPGA.

3.4.2. Introducción

En los sistemas de comunicaciones y particularmente en los dedicados a la codificación para el control de error y encriptamiento de datos se usan técnicas derivadas de la teoría de señales. Estas técnicas se aplican típicamente en la forma lineal debido a la simplicidad que ésto trae aparejado. Además cada una se implementa algorítmicamente o físicamente como una entidad independiente. Para cada sistema en particular se las elige con criterios de conveniencia práctica, y se las aplica en forma consecutiva o encadenada. La teoría de los sistemas no lineales [?, ?] aparece como un marco de trabajo ideal para ser utilizado en el contexto anteriormente mencionado. La existencia de los sistemas caóticos, y la relación de estos con la aleatoriedad, o pseudo aleatoriedad, otorga una plataforma de diseño que hasta hoy se encuentra poco explotada.

En los últimos veinte años se han presentado diversos trabajos que emplean caos en los sistemas de comunicaciones, como por ejemplo el empleo de portadoras caóticas sincronizadas en las transmisiones analógicas [?, ?]. Si nos centramos en la representación discreta un referente muy importante es el excelente trabajo de Kozic et als. [?, ?] en el que se presenta una técnica de modulación empleando mapas caóticos unidimensionales lineales por tramos, la técnica consiste en la introducción del mensaje a codificar en el bit menos significativo de la secuencia

generada. Obteniéndose una secuencia levemente alterada lo que impide que el sistema entre en ciclos periódicos.

En este trabajo se propone un grupo de atractores como generadores de señales pseudoaleatorias para realizar el proceso de codificación y encriptamiento. El esquema de codificación se basa en una familia de mapas cuadráticos bidimensionales, cuyas salidas presentan comportamiento caótico, con distintos atractores conforme a los coeficientes que se empleen. La idea es que cada palabra a codificar sea única con un juego de coeficientes que serán parámetros de un mapa cuadrático bidimensional. Como resultado de este procedimiento, la señal de salida son puntos pertenecientes a distintos atractores elegidos por la información a transmitir.

La ventaja de este método reside en que la estructura de toda la familia de mapas es única y común. Modificándose solamente los coeficientes se consiguen atractores distintos. Esta propiedad reduce y facilita la implementación en hardware. Resultados preliminares obtenidos mediante simulaciones muestran que el sistema presenta una performance comparable a la obtenida en sistemas clásicos de encriptamiento, en cuanto a probabilidad de error y a distancia mínima.

3.4.3. Mapas cuadráticos bidimensionales

En este trabajo se emplea una familia de mapas cuadráticos bidimensionales cuya estructura consta de dos ecuaciones cuadráticas en diferencias acopladas (Ec.3.4). La característica principal de este mapa es que los parámetros, a_i y b_i , son coeficientes reales y para ciertos valores de estos coeficientes la salida del sistema presentará comportamiento caótico.

Se obtiene así un subgrupo de mapas distintos con comportamiento caótico según la elección de estos coeficientes, en la Fig. 3.7 puede verse tres atractores para tres juegos distintos de valores de los coeficientes.

Estos mapas han sido ampliamente estudiados, por lo que la estabilidad, la sensibilidad a las condiciones iniciales y su dimensión fractal son parámetros ya conocidos [?, ?].

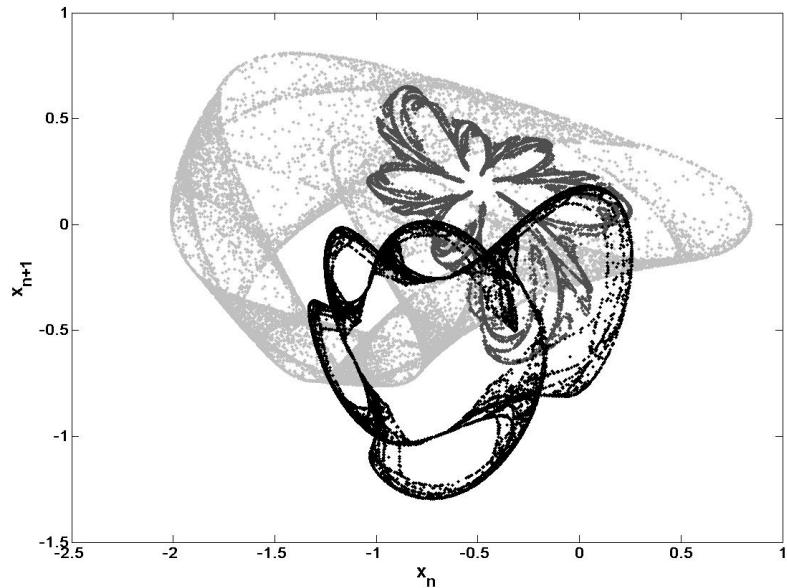


Figura 3.7: Atractores del sistema para tres juegos de coeficientes.

$$\begin{aligned}x_{n+1} &= a_0 + a_1 x_n + a_2 x_n^2 + a_3 x_n y_n + a_4 y_n^2 + a_5 y_n \\y_{n+1} &= b_0 + b_1 x_n + b_2 x_n^2 + b_3 x_n y_n + b_4 y_n^2 + b_5 y_n\end{aligned}\quad (3.4)$$

3.4.4. Implementación

Desde el punto de vista del esquema de codificación propuesto, estos mapas son muy atractivos por el hecho de contar con 12 coeficientes para generar cada atractor. Por lo tanto, las combinaciones posibles serán N^{12} , en donde N es la cantidad de símblos posibles según la aritmética utilizada. En nuestro caso empleamos una aritmética de 19 bits expresados en complemento a 2 con aritmética de punto fijo, con 1 bit de signo, 3 bits de parte entera y 15 bits de parte decimal. Esta aritmética limita y discretiza el plano xy que queda delimitado por $\Delta x = 4$, $-\Delta x = -4$, $\Delta y = 4$, $-\Delta y = -4$, como puede verse en la figura 3.7. Estas limitaciones al plano de atracción tienen como consecuencia dos cuestiones a tener en cuenta:

- Debido a que los coeficientes se generan con la misma aritmética que las variables, nos encontramos con $N = 2^{19}$ valores posibles para cada coeficiente, lo que arroja $(2^{19})^{12} \cong 4,3^{68}$ combinaciones posibles de coeficientes para generar distintos atractores.
- En cuanto a las trayectorias de los atractores sobre el plano discretizado, éstas se tornan periódicas debido a la discretización.

No todos los juegos de coeficientes generan atractores caóticos contenidos en el plano dado por la aritmética utilizada. Aunque esto no sería problema para la codificación/decodificación, se eligieron los coeficientes de modo que se generen atractores contenidos en el plano a modo de validación visual.

Dada la naturaleza de los mapas caóticos, un punto muy lejano a la zona de atracción puede hacer que el punto calculado para la próxima iteración diverja, por lo tanto las condiciones iniciales deben ser normalizadas antes de cambiar al mapa siguiente. Para solucionar este problema se utiliza la siguiente estrategia: Primero se define el plano mínimo que contiene al atractor. Para identificarlo se simularon los mapas mediante Quartus generando secuencias de salida lo suficientemente largas como para verificar la periodicidad. Luego se analizó este vector de datos con Matlab buscando los valores extremos en cada una de las variables: $X_{1\max}, X_{1\min}, Y_{1\max}, Y_{1\min}$. Estos límites delimitan al plano mínimo que contiene al atractor. La normalización dada por la ecuación 3.5 se aplica a la salida (x, y) para mapear este plano mínimo a todo el plano delimitado por la aritmética utilizada de dimensiones $\Delta x, -\Delta x, \Delta y, -\Delta y$. Segundo, se halla el plano máximo que contiene las condiciones iniciales que hacen que no diverja la solución sino que genere el atractor. Para esto se realizó un programa en Matlab que genera los atractores desde todas las condiciones iniciales del plano delimitado y discretizado por la aritmética utilizada, a continuación se marcan todos los puntos que generan trayectorias divergentes o bien convergentes a un punto fijo. Este proceso genera la zona de condiciones iniciales factible para generar atractores, nuevamente se identificaron los valores máximos y mínimos del área rectangular máxima que contenga todos sus puntos como condiciones iniciales factibles $X_{2\max}, X_{2\min}, Y_{2\max}, Y_{2\min}$. La normalización dada por la ecuación 3.6 se aplica a la entrada de condiciones iniciales (x_{n-1}, y_{n-1}) para mapear todo el plano de dimensiones $\Delta x, -\Delta x, \Delta y$ y $-\Delta y$ al de condiciones

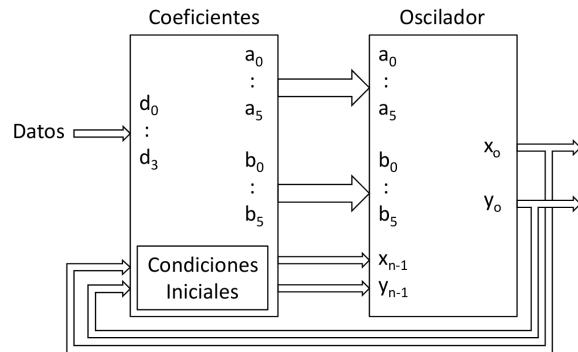


Figura 3.8: Generador de atractores.

iniciales factibles.

El problema de la existencia de puntos fijos para cierto conjunto de coeficientes y condiciones iniciales queda salvado al perturbar continuamente al atractor actual con valores afectados por la información.

Se generó un circuito en VHDL con un total de 16 juegos de parámetros seleccionables con la palabra de entrada de 4 bits que se desea encriptar. Esta palabra multiplexa estos coeficientes y alimenta un oscilador que calcula la próxima iteración de datos, además, este circuito almacena la salida del oscilador y la realimenta como “condición inicial” para calcular la iteración siguiente (Fig. 3.8). Como resultado de este proceso, la salida encriptada resulta ser el oscilador actual seleccionado por la palabra de entrada perturbado por la historia de los mapas seleccionados por las entradas anteriores. Este circuito de dos bloques se encarga de generar los atractores, por lo que se lo llama “generador de atractores”.

Para la primer iteración, las condiciones iniciales son $(x; y) = (0,1; 0,1)$ para cualquiera de los atractores.

$$\begin{aligned}
 x_{1norm} &= a_{1x}x + b_{1x} \\
 y_{1norm} &= a_{1y}y + b_{1x} \\
 a_{1x} &= \frac{2\Delta x}{x_{1max} - x_{1min}} \\
 a_{1y} &= \frac{2\Delta y}{y_{1max} - y_{1min}} \\
 b_{1x} &= -\frac{x_{1max} - x_{1min}}{2} \\
 b_{1x} &= -\frac{y_{1max} - y_{1min}}{2}
 \end{aligned} \tag{3.5}$$

$$\begin{aligned}
 x_{1norm} &= a_{2x}x + b_{2x} \\
 y_{1norm} &= a_{2y}y + b_{2x} \\
 a_{2x} &= \frac{x_{2max} - x_{2min}}{2\Delta x} \\
 a_{2y} &= \frac{y_{2max} - y_{2min}}{2\Delta y} \\
 b_{2x} &= \frac{x_{2max} - x_{2min}}{2} \\
 b_{2x} &= \frac{y_{2max} - y_{2min}}{2}
 \end{aligned} \tag{3.6}$$

Codificador

El bloque del Codificador consiste en circuito generador y acondicionamiento de la salida. Para codificar una palabra de cuatro bits de entrada se generan los valores de x e y con el circuito generador correspondiente a esta palabra y se los concatena en un circuito posterior formando un vector $[x : y]$ (Fig. 3.9). De esta forma cada palabra de información a ser enviada será representada por la salida xy del oscilador del atractor correspondiente, por lo tanto una palabra a codificar no se corresponderá con una palabra codificada, dos palabras iguales generarán dos salidas distintas.

Decodificador

Un segundo circuito generador de atractores funciona en el decodificador generando las 16 palabras posibles para la próxima iteración. Luego, se ingresan

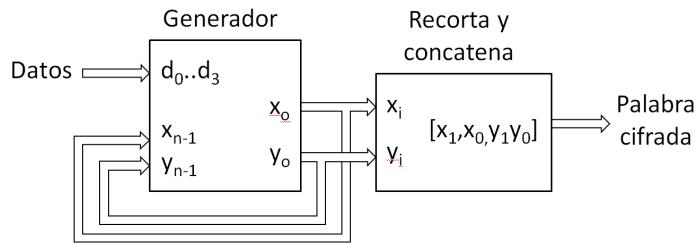


Figura 3.9: Codificador.

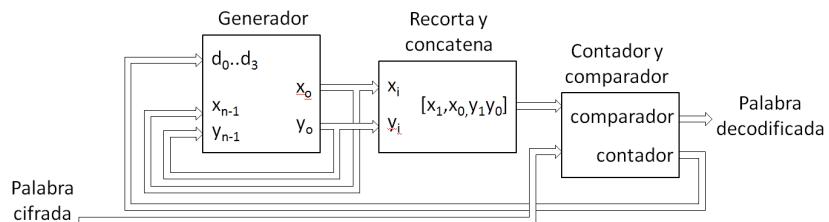


Figura 3.10: Decodificador.

todas estas posibles palabras cifradas junto con la que se desea decodificar a un comparador que aplica una XOR a la palabra ingresada contra todas las palabras posibles generadas localmente para decodificarla. La salida de este circuito es la palabra decodificada (Fig. 3.10).

3.4.5. Resultados

Se realizó un primer esquema del diseño mediante la herramienta Quartus II v8.0 de ALTERA, para implementar el sistema en una FPGA *Altera Cyclone III EP3C120*.

Resultados de la compilación del codificador pueden evrse en la tabla XXX. El esquema del codificador se muestra en la figura XXX.

Se obtuvieron resultados preliminares de simulaciones realizadas mediante el programa Matlab y mediante simulaciones con el programa Quartus de Altera, estas últimas tienen en cuenta el empleo de la precisión finita elegida para representar los valores.

En la Fig. 3.11 se pueden ver las salidas del bloque generador para una transmisión de los datos [1,2,3,2,3,3,1,3,1,3,1]. En este caso se mantiene el dato

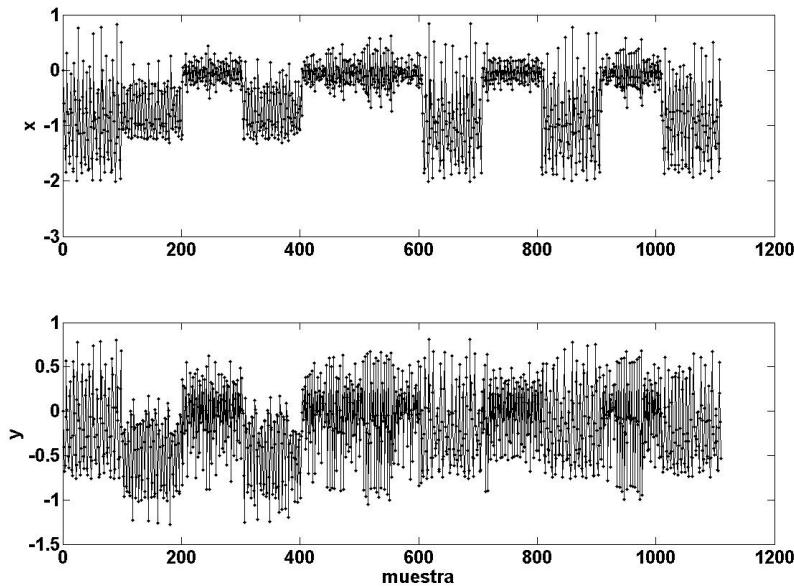


Figura 3.11: Señales a transmitir.

a enviar durante 100 ciclos con el objetivo de que sea visible en la figura, en el sistema real cada oscilador codifica una palabra de información en cada iteración. Aquí puede observarse que el sistema cambia el atractor generado según los coeficientes que dependen de la entrada de información a transmitir.

En cuanto al análisis de performance que presenta el sistema se deben tener en cuenta dos aspectos:

- La distancia mínima de la modulación codificada resultante. Esta es usualmente empleada para proveer un límite de error en la región de piso.
- Una descripción precisa de la tasa binaria de error del sistema o BER (en inglés, Bit Error Rate) también es un parámetro muy importante, ya que da una estimación del comportamiento que presentara el código.

3.4.6. Conclusiones

Se desarrolló un nuevo método para el diseño de sistemas de criptocodificación mediante el empleo de mapas caóticos cuadráticos acoplados. Se obtuvieron

resultados muy prometedores hasta el momento mediante las simulaciones realizadas. También se obtuvieron buenos resultados en la implementación en VHDL del sistema codificador, en la que se verificó que la salida generada por el sistema no cayera en ciclos periódicos debido a la utilización de presición finita.

3.5. Implementación de atractor Determinístico - Estocástico

3.5.1. Introduction

En los últimos treinta años, los sistemas caóticos han producido una revolución en nuestra visión de la naturaleza ya que tienen dos características contrastantes: (1) son deterministas porque un modelo matemático determina su dinámica, pero (2) debido a su sensibilidad a las condiciones iniciales, se pierde la predicción a largo plazo y, en consecuencia, pueden incluirse en la clase de sistemas estocásticos que se estudian mediante herramientas estadísticas. En realidad, si los sistemas caóticos pudieran implementarse con una precisión infinita, serían deterministas en sentido estricto, pero la precisión infinita es un ideal imposible de alcanzar en la electrónica digital.

Esta *dualidad determinista-estocástica* hace que los sistemas caóticos sean especialmente interesantes para las aplicaciones de ingeniería, en la medida en que las señales generadas pueden usarse como ruidos controlados en una amplia gama de aplicaciones. Sin embargo, las secuencias caóticas realmente presentan correlaciones internas no lineales, lo que las hace inadecuadas para ser utilizadas como "buenas" PRNG. Es necesario utilizar técnicas de aleatorización para mejorar la aleatoriedad de la serie [?].

En aplicaciones digitales, el tiempo y la variable de estado tienen valores discretos. La discretización de tiempo impone el uso de un algoritmo para aproximar las ecuaciones diferenciales de tiempo continuo que modelan el sistema. El algoritmo más simple es el método de Euler de primer orden en el que los diferenciales se reemplazan directamente por incrementos finitos. Los algoritmos más elaborados, como los algoritmos de paso variable Runge-Kutta de cuarto orden (o superior), hacen que el sistema discreto evolucione más cerca del sis-

*3.5. IMPLEMENTACIÓN DE ATRACTOR DETERMINÍSTICO - ESTOCÁSTICO*69

tema continuo, pero con mayores requisitos de recursos de hardware y tiempos de cálculo. En consecuencia, deben usarse solo si la exactitud es un requisito de la aplicación específica. Este no es el caso en PRNG, donde la aleatoriedad es la característica principal que debe garantizarse.

Para hacer los cálculos aritméticos, las computadoras deben representar internamente las señales por un número finito de bits, esto significa que los valores se describen usando aritmética de precisión finita. Esta restricción puede ser crítica para un sistema caótico ya que es extremadamente sensible a la aritmética empleada e incluso puede perder su comportamiento caótico.

En este capítulo, implementamos un oscilador discreto de Lorenz obteniendo mediante el uso del algoritmo de primer orden de Euler y tres estándares de representación diferentes. Además aplicamos técnicas de aleatorización a las variables de estado para obtener un PRNG en tiempo real. El objetivo de este trabajo es estudiar la influencia del procedimiento de discretización en la dinámica del sistema. En [?] se estudió el grado de estocasticidad de un sistema caótico determinista mediante su implementación en coma flotante de precisión simple (32 bits).

La determinación del grado de estocasticidad tiene como objetivo proporcionar una metodología de diseño optimizada para la utilidad prevista dada al sistema caótico. Así, por ejemplo, hay aplicaciones que requieren que el sistema caótico reemplace un sistema estocástico (criptografía [?], generadores de secuencia para difundir comunicaciones de espectro [?, ?], generadores de números pseudoaleatorios [?, ?, ?], reducción de la interferencia electromagnética [?], etc.). Por otro lado, algunas aplicaciones requieren previsibilidad a largo plazo, por ejemplo para reproducir el sistema caótico de la manera más precisa posible, este es el caso de las comunicaciones analógicas que usan portadores de sincronización caóticos [?, ?].

Otro problema es la determinación exacta del período de la secuencia pseudoaleatoria. Para generadores que se basan en operaciones lineales, el problema se ha estudiado en profundidad y existen criterios de diseño bien conocidos para obtener dispositivos de ciclo máximo. Ejemplos de algoritmos lineales son: el algoritmo Mersenne Twister que es un generador de números aleatorios muy rápido de período $T = 2^{19937} - 1$ [?] y Multiply-With-Carry (MWC), un méto-

do inventado por George Marsaglia para la generación de secuencias de enteros aleatorios basados en un conjunto inicial de dos a muchos miles de valores de semilla elegidos al azar, presenta períodos inmensos, que van desde alrededor de 2^{60} a $2^{2000000}$ [?]. Sin embargo, desde el punto de vista criptográfico son débiles. Cuando se trata de aplicaciones criptográficas, no se recomiendan métodos lineales para generar secuencias pseudoaleatorias (como LFSR, LCG o sus combinaciones adecuadas), ya que algoritmos eficientes para predecir la secuencia sobre la base de una secuencia de observación relativamente corta están a disposición [?, ?]. Por otro lado, para la mayoría de las familias de generadores no lineales, el problema parece ser intratatable y, con pocas excepciones, no suele existir una evaluación analítica de sus períodos [?].

Elegimos el sistema caótico de Lorenz porque es un sistema ampliamente estudiado y también ha sido implementado por otros autores con diferentes metodologías cite Asseri2002, Azzaz2009, Azzaz2010. Por ejemplo, en cite Asseri2002, se implementó un oscilador caótico Lorenz mediante una caja de herramientas del Generador de Sistemas Xilinx que funciona bajo *MATLAB – Simulink* *copyright*. Esta caja de herramientas convierte el modelo MATLAB-Simulink en el modelo de Xilinx System Generator. Luego se obtiene el código VHDL. Como es una operación automática, no permite ciertos cambios específicos y presenta algunas limitaciones. La operación de integración se aproximó con el algoritmo de Euler, utilizando bloques de suma y retardo. La implementación propuesta en cite Azzaz2009 y cite Azzaz2010 usa *RK4* en una arquitectura de punto fijo de 32– bit. La aplicación considerada fue la generación de clave aleatoria caótica para el cifrado de flujo de datos.

In this paper *Quartus II 7,2[©]* development software is used to generate the VHDL hardware description language and the physical implementation is made on the development board *Altera[©] Cyclone III EP3C120*. Three representations are studied: 1) floating-point IEEE 754 standard, 2) decimal fixed point, and 3) integer arithmetics [?]. Each representation involves a different number of bits. We consider two floating point representations for the standards single or double, with 32 and 64 bits respectively to represent the sign, the exponent and the mantissa; decimal fixed point arithmetics uses p bits to represent the integer part and m bits to represent the fractional part. We considered 32 and

64 bits fixed point representations with 9 bits for the integer part and 1 bit for the sign and the remaining 22 or 54 bits for the fractional part. In the k bits integer arithmetics the alphabet has 2^k symbols. We considered $k = 54$.

En esta sección se utilizó el software *QuartusII[®] 7.2* para generar el lenguaje de descripción de hardware VHDL y la implementación física se realiza en la placa de desarrollo *Altera[®] Cyclone III EP3C12*. Tres representaciones numéricas son estudiadas: 1) punto flotante IEEE 754 estándar, 2) punto fijo decimal, y 3) aritmética entera [?]. Cada representación implica una cantidad diferente de bits. Consideramos dos representaciones de coma flotante para los estándares simple o doble, con 32 y 64 bits respectivamente para representar el signo, el exponente y la mantisa; La aritmética decimal de punto fijo usa p bits para representar la parte entera y m bits para representar la parte fraccional. Consideramos representaciones de punto fijo de 32 y 64 bits con 9 bits para la parte entera y 1 bit para el signo y los 22 o 54 bits restantes para la parte fraccionaria. En los aritméticos enteros k bits el alfabeto tiene 2^k símbolos. Consideramos $k = 54$.

Para cuantificar la aleatoriedad del sistema, se utiliza el conjunto de pruebas DIEHARD de Marsaglia. Estas pruebas han sido ampliamente utilizadas en la literatura abierta y son muy efectivas para clasificar sistemas determinísticos y estocásticos.

3.5.2. Discretización temporal del oscilador de Lorenz

El sistema de Lorenz se define mediante el siguiente conjunto de ecuaciones diferenciales ordinarias acopladas:

$$\begin{aligned} \frac{dx}{dt} &= -\delta(x - y), \\ \frac{dy}{dt} &= \Gamma x - y - xz, \\ \frac{dz}{dt} &= -bz + xy, \end{aligned} \tag{3.7}$$

donde δ , Γ y b son parámetros constructivos del sistema. Para ciertos valores de estos parámetros, el sistema tiene un comportamiento caótico. Siempre se requiere un algoritmo para convertir un sistema dinámico continuo en un sistema dinámico de tiempo discreto. El algoritmo más simple fue propuesto por Euler

y, para las Ecs. ?? surge el siguiente modelo discreto:

$$\begin{aligned}\tilde{X}_{t+\Delta t} &= \tilde{X}_t + \Delta t \left[-\delta (\tilde{X}_t - \tilde{Y}_t) \right], \\ \tilde{Y}_{t+\Delta t} &= \tilde{Y}_t + \Delta t \left[-\tilde{X}_t \tilde{Z}_t + \Gamma \tilde{X}_t - \tilde{Y}_t \right], \\ \tilde{Z}_{t+\Delta t} &= \tilde{Z}_t + \Delta t \left[\tilde{X}_t \tilde{Y}_t - b \tilde{Z}_t \right],\end{aligned}\quad (3.8)$$

donde Δt es el tamaño de paso de tiempo y \tilde{X} , \tilde{Y} y \tilde{Z} son variables de estado de tiempo discreto (reales).

El algoritmo de Euler es un algoritmo de un solo paso porque para calcular las variables en el momento $t + \Delta t$, solo es necesario conocer los valores en el instante anterior. Calculando iterativamente con un paso apropiado Δt , es posible obtener la evolución del sistema discreto. Es razonable esperar que cuanto menor sea el valor de Δt , más exactos serán los valores obtenidos. Aunque, al reducir el valor de Δt se incrementa la cantidad de cálculos y esto genera más errores de redondeo.

En aplicaciones que requieren una reproducción exacta de la dinámica del sistema continuo, los algoritmos más exactos son obligatorios, pero en el caso de los PRNG, solo las propiedades estadísticas y la aleatoriedad de las series temporales son importantes y, por consiguiente, el algoritmo de Euler es lo suficientemente bueno.

3.5.3. Discretización de las variables de estado

As pointed out in the introduction three different numerical representations are used in this paper. Each one is described in the following subsections.

IEEE 754 standard implementation

Floating point representation is one of the methods to represent real numbers with finite precision. The advantage of floating-point representation over fixed-point and integer representations is it can support a much wider range of values because it automatically scales each number to use the full word length for the mantissa; this is done by moving the decimal point (this procedure implies a change in the value of the exponent) towards the position of the most significant bit. Consequently full precision is preserved even for small numbers. Binary

3.5. IMPLEMENTACIÓN DE ATRACTOR DETERMINÍSTICO - ESTOCÁSTICO73

floating-point arithmetic is best suited to work with real-world quantities over a wide range of scales. Special care must be taken because of some accuracy issues.

The single precision IEEE 754 standard assigns 23 bits to the mantissa (bit 0 to 22), the exponent occupies the following 8 bits (bit 23 to 30) and the 31-st bit is assigned to the sign. The double precision IEEE 754 standard assigns 52 bits to the mantissa (bit 0 to 51), the exponent occupies the following 11 bits (bit 52 to 62) and the 63-nd bit is assigned to the sign.

Floating point arithmetic operations are more complicated than fixed point ones. Their execution requires more time and complex hardware. However, thanks to technological advance and the development of new materials, nowadays there exists FPGAs with more memory and resources, capable of working at high frequencies in these standards.

Fixed point implementation

When all the numbers are within a known range it is possible to achieve more accuracy using the so-called fixed point representation instead of the floating point representation. The hardware required to manipulate these representations is the same commonly used to perform integer operations and it is less expensive than the one required for the floating-point case.

To avoid the overflow it is necessary to initially perform an analysis to determine the largest value involved in the calculus, including the intermediate operations. With this information the minimum number of bits to be employed is determined. Once established the number of bits required to represent the integer part, one additional bit is used to represent negative numbers (based two complement). The remaining bits added are used to improve the precision as they represent the decimal part.

The addition, subtraction and multiplication operations are implemented in the same way as in integer arithmetic. It is only necessary to take care of the position of the radix point.

In this paper we considered two cases, 32 and 64 bits for each whole number. In both cases we used 9 bits for the integer part, plus 1 bit for the sign, leaving the remaining bits, 22 or 54 respectively, for the decimal part.

Integer arithmetics implementation

In integer arithmetics the circuitry can be significantly reduced if power of 2 divisors are adopted. To obtain the integer version for the Lorenz system the following transformations of polarization and scaling, were done [?]:

$$\begin{aligned} X_t &= (\tilde{X}_t + B) S, \\ Y_t &= (\tilde{Y}_t + B) S, \\ Z_t &= (\tilde{Z}_t + B) S, \end{aligned} \quad (3.9)$$

where B y S are the polarization and scale parameters respectively. Replacing eq. (3.9) in (3.8) it is obtained:

$$\begin{aligned} X_{t+\Delta t} &= X_t + \Delta t \delta (Y_t - X_t) \\ Y_{t+\Delta t} &= (1 - \Delta t) Y_t + \Delta t B + \Gamma X_t + \Delta t B Z_t \\ &\quad - \frac{\Delta t}{S} X_t Z_t + \Delta t B S (1 - \Gamma - B), \\ Z_{t+\Delta t} &= (1 - \Delta t b) Z_t - \Delta t B (X_t + Y_t) \\ &\quad + \frac{\Delta t}{S} X_t Y_t + \Delta t B S (B - b). \end{aligned} \quad (3.10)$$

In this case it was adopted: $\delta = 8$, $\Gamma = 24$, $b = 2$, $\Delta t = 2^{-n}$, $B = 40$, $S = 512$.

Some care must be taken when the system parameters are chosen, in this case integer coefficients were selected and an analysis of stability was done in order to ensure that the system do not converge to a fixed point or to a low

period orbit. The final system is:

$$\begin{aligned}
 X_{t+\Delta t} &= X_t + \text{floor} \left[\frac{Y_t}{2^{n-3}} \right] - \text{floor} \left[\frac{X_t}{2^{n-3}} \right] \\
 Y_{t+\Delta t} &= Y_t - \text{floor} \left[\frac{Y_t}{2^n} \right] + \text{floor} \left[\frac{X_t}{2^{n-6}} \right] \\
 &\quad + \text{floor} \left[\frac{Z_t}{2^{n-3}} \right] + \text{floor} \left[\frac{Z_t}{2^{n-5}} \right] \\
 &\quad - \text{floor} \left[\frac{X_t}{2^{(22+\text{floor}[\frac{n}{2}+1])}} \right] \\
 &\quad \text{floor} \left[\frac{Z_t}{2^{(22+\text{floor}[\frac{n}{2}])}} \right] - 2^{(44-n)} 2520 \\
 Z_{t+\Delta t} &= Z_t - \text{floor} \left[\frac{Z_t}{2^{n-1}} \right] - \text{floor} \left[\frac{(X_t + Y_t)}{2^{n-3}} \right] \\
 &\quad - \text{floor} \left[\frac{(X_t + Y_t)}{2^{n-5}} \right] + \text{floor} \left[\frac{X_t}{2^{(22+\text{floor}[\frac{n}{2}+1])}} \right] \\
 &\quad \text{floor} \left[\frac{Y_t}{2^{(22+\text{floor}[\frac{n}{2}])}} \right] + 2^{44-n} 1680
 \end{aligned} \tag{3.11}$$

This discrete system has a chaotic behavior (in fact pseudochaotic) and all the divisors are a power of 2. All the preprocessing procedure of the equations minimizes the required hardware resources (as it will be shown later).

3.5.4. Some issues on chaotic PRNG

PRNGs are widely used in physics and engineering. Some applications are: Monte Carlo simulations [?], cryptography [?], communications theory [?] and some aspects of nanotechnology [?].

In some applications it is required that the chaotic system replaces an stochastic system. This is the case for example in masking applications [?], generation of sequences for spread spectrum technique [?, ?] and electromagnetic compatibility improvement [?], pseudo-random number generators [?, ?, ?], etc.

Chaotic dynamical systems have been employed as PRNGs [?, ?, ?], because they are able to generate stochastic-like signals out of underlying simple models that are easy to implement via appropriate software or hardware. Usually, a suitable manipulation of the time-series that these models generate is required to improve their statistical properties.

To eliminate or mitigate the inner correlational structures that are not

desirable here, two solutions, that do not require a larger hardware, are analyzed:

1. *discarding*: form a new sequence whose elements are integer numbers formed by the 32 less significant bits of each data item (named x_{disc} , y_{disc} and z_{disc} respectively);
2. *concatenating*: form new 32 bits integers by concatenating the least significant bits of each variable (11 bits of z , 10 bits of y and 10 bits of x , note that this is one of many possibilities). Named zyx .

These procedures were applied to the output sequences generated by all the implementations described in the previous sections. In addition, we varied Δt to find its optimal value.

There are several basic properties any good PRNG must fit: long cycle length, randomness, speed, reproducibility and portability. Several test suites [?] are readily available to researchers in academia and industry who wish to analyze their newly developed PRNG. Some general purpose test suites are DIEHARD by George Marsaglia [?], Crypt-XS by Helen Gustafson of the Queensland University of Technology [?], the National Institute of Standards and Technology (NIST) statistical test suite [?], Test U01 by L'Ecuyer and R. Simard [?] and DIEHARDER [?]. In this paper we used the 15 most stringent tests of DIEHARD [?] to measure the stochasticity of each implementation but if the specific application is a PRNG the use of all the above mentioned tests is recommended, specially NIST 800/22 and Test U01.

For each PRNG a file with more than 80×10^6 bits is required. Each run of each test in DIEHARD returns a p -value, which should be uniform on $[0, 1)$ if the input file contains truly independent random bits. Those p -values must be $p < 0,025$ or $p > 0,975$ for us to consider the test has been passed. Each test runs several times giving a p value for each run. Combining all these p -values (229 for each PRNG) we obtained an overall p -value by means of the KStest. Only if all the individual p 's and also the global p are in the range pointed above we put "yes" in Table ??.

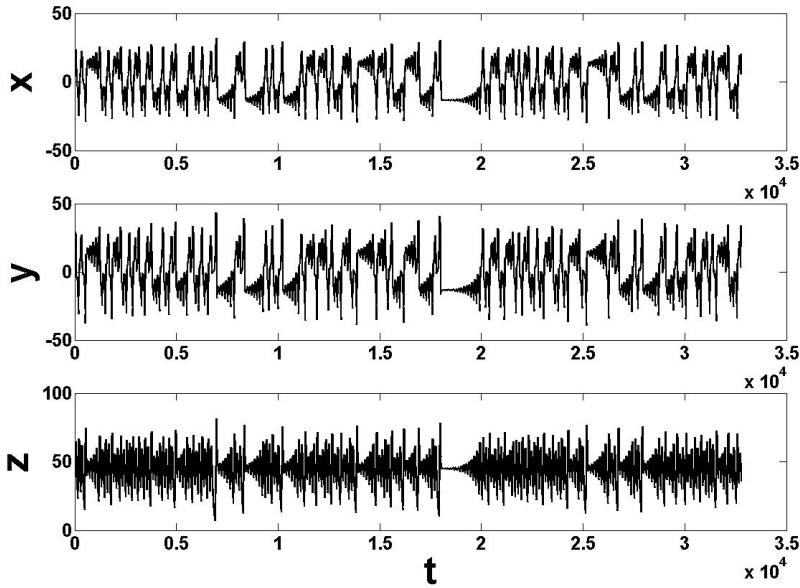


Figura 3.12: Lorenz time series.

3.5.5. Results

The implementations presented here were fully developed with *Quartus II 7.2[©]* software. The physical implementations were made on *Altera[©] Cyclone III EP3C120* development kit.

SignalTap II Embedded Logic Analyzer was used to make the hardware evaluation for each design. This is a system-level debugging tool, provided by *Altera*, that captures and displays the real-time signal behavior. It allows one to observe interactions between hardware and software in system designs. After capturing the data and saving it to a *SignalTap II* file, it can be analyzed and viewed as a waveform [?].

Figs. 3.12 and 3.13 show respectively the temporal series and attractor, obtained by the hardware implementation with $\Delta t = 0,0045$ and floating point single precision (figures with the other numerical representation are very similar).

In Table ?? a comparison between the compilation results in the three numeric representations studied here:

- *floating point arithmetics*: two cases, single and double precision (Float(32bits)

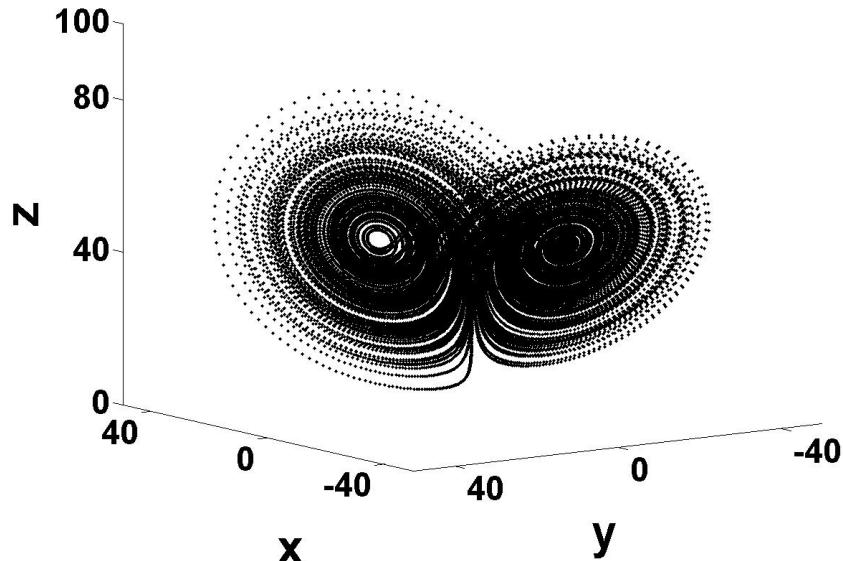


Figura 3.13: Lorenz attractor.

and `Float(64bits)` respectively),

- *integer arithmetics* (`Integer(54bits)`) and
- *fixed point arithmetics*: two cases, both with 9 bits for the integer part plus 1 bit for the sign plus 22 bits (`Fixed point(32bits)`) or 54 bits (`Fixed point(64bits)`) respectively, for the fractional part.

The hardware required is shown in Table ???. The integer arithmetic implementation is the one that employs minimum resources and supports a higher f_{max} , the reason for this is that the equations implemented were previously optimized for this particular application. In the case of floating point representation optimization was made to diminish the area but other optimization techniques may be applied to improve frequency, or power consumption [?, ?].

In order to employ this system as a hardware PRNG output data are processed with *discarding* and *concatenating* techniques. Both techniques keep the least significant bits because they present the more variable behavior. In the case of *concatenating* technique, the noisiest part of each state variable is kept and recombined, and a 32-bits sequence output is obtained in each iteration.

3.5. IMPLEMENTACIÓN DE ATRACTOR DETERMINÍSTICO - ESTOCÁSTICO79

Cuadro 3.1: Compilation results CYCLONE III EP3C120F780C7.

	Fixed point(32bits)	Fixed point(64bits)	Float(32bits)	Float(64bits)	In
Total logic elements	2,392	6,104	8,176	17,532	
Percentage of logic elements [%]	2,00	5,12	6,86	14,72	
Total registers	1,658	1,754	4,753	8,532	
clk f_{max} [MHz]	37,82	20,51	7,48	5,42	
Throughput [Mbs]	1,210,24	656,32	14,96	173,44	

Cuadro 3.2: DIEHARD tests results.

	$\Delta t = 1/2^n$	0,015625	0,0078125	0,00390625	0,001953125	0,0009765625
Float (64 bits)	x_{disc}	no	no	yes	yes	no
	y_{disc}	no	no	yes	yes	no
	z_{disc}	yes	yes	yes	no	no
	zyx	no	no	no	no	yes
Fixed Point (64 bits)	x_{disc}	yes	yes	no	no	no
	y_{disc}	yes	yes	no	no	no
	z_{disc}	yes	yes	no	no	no
	zyx	yes	yes	yes	no	no
Integer (54 bits)	x_{disc}	yes	yes	no	no	no
	y_{disc}	yes	yes	no	no	no
	z_{disc}	yes	yes	no	no	no
	zyx	yes	yes	yes	yes	yes

For the stochasticity analysis data files with 3,000,000 32-bits words each were generated for $\Delta t = 2^{-n}$, with $n = 6,7,8,9$ and 10. DIEHARD tests were calculated for all the files generated. In Table ?? some of the most relevant results are reported. x_{disc} , (y_{disc} , z_{disc}) means x -time series (y , z) after the discarding randomization technique is applied. xyz means the time series obtained by means of the *concatenating* randomization technique. This Table ?? shows that in the cases of integer and fixed point implementations the discarding randomization technique works better as Δt increases because, for lower Δt consecutive elements in the time series are more correlated and this randomization technique does not mix them enough. In order to use lower values of Δt more bits should be discarded to obtain good PRNGs. On the other hand the concatenating randomization technique performs well regardless of the value of Δt (within the analyzed range).

3.5.6. Conclusions

From the results presented herein, it is possible to conclude that to obtain a PRNG optimum results correspond to the integer arithmetics representation in both hardware (resources and frequency) and statistical properties. The *concatenating* randomization technique makes the quality independent of Δt for this numeric representation. For *discarding* randomization technique good results are obtained only for big values of Δt . The same happens for fixed point implementations with both randomization techniques.

In terms of usage resources and frequency limitations integer arithmetics performance is considerable better than floating point and fixed point. Let us remark that to minimize resources preprocessing of the chaotic system is required (scaling and polarization) to get dividers that are a power of 2, as explained in subsection 3.5.3.

On the other hand, for the floating point arithmetic's case the exponent is discarded in all the used randomization techniques and consequently the dynamics is highly perturbed by the randomization process. Then, as shown in Table ?? Δt is not the relevant variable to predict a good or a bad PRNG.

3.6. Stochastic Degradation of the Fixed-point version of 2D-Chaotic Maps

3.6.1. Introduction

Chaotic systems have an increasing number of applications and their implementation is specially involved due to the extreme sensitivity to initial conditions. In general, these systems are used for the generation of controlled noises, these digital pseudo-random noise generators (PRNGs) can be employed in a large number of electronic applications, such as encryption sequences for privacy, multiplexing techniques, electromagnetic compatibility [?, ?, ?, ?, ?]. In computers and digital devices only *pseudo chaotic* attractors can be generated. But discretization may destroy the *pseudo chaotic* behavior and consequently is a nontrivial process.

Several strategies have been proposed in the literature for a correct selection

of the optimal number of bits in hardware implementations. However, most of these procedures are limited to linear systems [?, ?]. In digital chaotic systems, a completely different behavior may be obtained by varying the precision. This issue has gained interest recently, and several new schemes have been proposed [?, ?, ?].

In short, in spite of the arithmetic used, i. e. fixed-point or floating-point arithmetic, the set of numbers that can be represented is limited. Even using extremely high precision as done by Liao and Wang [?] the generated sequences by a chaotic system using digital hardware will always be periodic.

Grebogi's work [?] showed that the average length T of periodic orbits of a dynamical system implemented in a computer, scales as a function of the computer precision ξ and the correlation dimension of the chaotic attractor, as $T \sim \xi^{-D/2}$. In [?] some findings on a new series of dynamical indicators, which can quantitatively reflect the degradation effects on a digital chaotic map realized with a fixed-point finite precision, have been reported, but they are restricted to 1D piecewise linear chaotic maps (PWLCM). In [?] the effect of numerical precision on the mean distance and on the mean coalescence time between trajectories of deterministic maps with either multiplicative noise parameter or with an additive noise term was investigated. Nepomuceno et al. [?] studied the changes in the pseudo orbits of continuous chaotic systems when varying the time and discretization schemes.

Liao [?, ?, ?] proposed a numerical algorithm, namely the clean numerical simulation (CNS). He claimed that the CNS gives an extremely precise numerical approach for chaotic dynamic systems in a given finite interval. If the initial condition were exact, then the long-term prediction of chaos would be possible in theory, unfortunately, the required CPU time increases exponentially as the number of digits precision and Taylor expansion order increases (for continuous systems), so that it is practically impossible to give true trajectories of chaos in a very long interval. Unlike our analysis, which is carried out on a map, we do not address the issue arising from the time discretization. What is more, in [?], lower precisions are despised because they are very far from the real trajectory. In contradistinction our approach comes from the hardware implementation point of view, where using minimum resources is mandatory, does not dismiss

any precision. Instead, our goal is to investigate the characteristics for each precision so that the designer has a complete overview of the options to be used in its implementation. So that designers will be able to decide which properties to rescind according to the available resources and requirements.

One important thing to note is that besides analyzing the changes in the period lengths, the statistical properties of the sequences will be different from those of the real system and so they also should be analysed. In [?] an excellent work about the consequences finite precision has on the periodicity of a PRNG based on the logistic map was developed. There, the number, delay, and period of the orbits of the logistic map at varying degrees of precision were determined, however they lacked a statistical analysis. Our research complements their work by adding statistical quantifiers. What is more, they analysed the floating-point architecture of the map, while here we have chosen fixed-point architecture as it is the optimal architecture for hardware implementations. From an engineering point of view, fixed-point arithmetic is more efficient than floating-point, it consumes fewer resources and their operations require lower number of clock cycles. As a consequence, power consumption is also diminished.

Among many chaotic systems available in the literature, we are interested in a family of $2D$ -maps proposed by Sprott [?]. The main characteristic of this system is it presents multiple chaotic attractors depending on the selected point in the parameter's space, this feature is very attractive to be used in electronic applications. Only results for the analytical representation of the maps in [?] have been published in the open literature.

The objective of this paper is to extend the analysis to the digital version, to make possible the hardware implementation in fixed-point arithmetic. For which it is imperative to know both characteristics, period length and degree of randomness, of the sequences. We developed a detailed analysis of the *degradation* of the multiaffector chaotic system as a fixed-point implementation is used. By *degradation* we mean: (a) the appearance of stable fixed points and stable periodic orbits with short periods, inside a floating-point domain of attraction without stable orbits; (b) the attractor itself becomes periodic and its statistical characteristics change, making the system more deterministic.

The main contributions of this paper are:

- the analysis of the domains of attraction of the chaotic attractors for a given set of parameters as the number of bits increases; in terms of period lengths and the appearance of stable fixed points and periodic orbits with short periods are specially considered;
- the determination of the consequent *threshold width* for the bus, in order to make the statistical properties of the digital implementation close to those of the floating-point implementation;
- two different probability distribution functions (PDF) are assigned to evaluate the stochasticity of the time series for different bus widths. Each PDF P is measured by the respective normalized Shannon entropy $H(P)$. These entropies have abrupt changes at specific bus widths. Period's lengths and *MLE* are also evaluated and results are compared with H_s .

3.6.2. Preliminary Concepts

When iterating chaotic maps in \mathbb{R}^2 , after a transient that depends on the mixing parameter (r_{mix}), the generated sequence limits in a point or a collection of points called an attractor. A chaotic map can have one or more attractors. Attractor domain is called to all the initial conditions (ICs) that converge to each attractor. The ergodic sequences of the attractors, generated by the map, have a determined distribution called Invariant Probability Density Function (IPDF). Main characteristics of chaotic maps, IPDF and r_{mix} , can be obtained by calculating the Frobenius-Perron operator (FPO), which depends on the map's structure. The fixed points of its spectrum are the invariant densities and they correspond to the eigenvectors with eigenvalue equal to one, the mixing constant corresponds to the second largest eigenvalue of the FPO, [?, ?].

When using finite precision, this analysis is not valid, all attractors take the form of fixed points or periodic orbits. The FPO of the map no longer describes the sequences' characteristics. Regarding the attractor domain, it will also change when digitalized, each initial value will be part of, or will converge to, a certain fixed point or periodic orbit. Generally, many new periodic orbits appear, and change when the number of bits employed varies.

With the purpose of utilizing these systems in electronic applications it be-

comes necessary to understand how the attraction domain evolves with the variation of bits employed. It is mainly important to know which is the period's length and the *randomness degree* of the cycle at which each seed converges. For this reason, we have included randomness quantifiers that indirectly estimate a sort of r_{mix} and IPDF of the digitalized system.

In this paper we have emulated the behaviour of a digital hardware implementation, such as FPGA, Complex Programmable Logic Device (CLPD) or Application Specific Integrated Circuit (ASIC), to exactly replicate the operation of the device. Our interest is to measure how the domains of attraction degrade with a change in the number of bits n employed, as well as to find the threshold value n_{min} .

3.6.3. Results

An ANCI C code that simulates iterating a nonlinear system, the quadratic map, in any digital electronic device was developed in order to generate sequences which were then analyzed.

It iterates the 2D-quadratic map 10^5 times, in this case coefficients a_0 to a_{11} have the values: $\{a_i\} = \{-1,0, 0,9, 0,4, -0,2, -0,6, -0,5, 0,4, 0,7, 0,3, -0,5, 0,7, -0,8\}$. The system was intended to be working in fractional fixed-point architecture with n bits, where $n = n_i + n_f$, in two's complement representation (Ca_2). In this case we employed $n_i = 4$ bits for representing the integer part, and the code automatically varies the number of bits representing the fractional part of the number, n_f , in order to analyze how the system reacts when the precision changes. The code runs through all the ICs within the interval $[-2, 2]$ in steps determined by the current n_f , so, the step_grid will be:

$$step_grid = \frac{1}{n_f, 2^{n_f}}. \quad (3.12)$$

On each case it was determined whether the systems evolves to a fixed point, diverges or goes towards a periodic cycle, also sequences for each cycle of that IC using n_f bits of precision were generated. These data were then evaluated using the randomness quantifiers previously introduced in Section ??.

Figure 3.14 displays the obtained domains of attraction for $n_i = 4$ and some

values of n_f . The abscissa and ordinate axis correspond to initial values of x and y respectively. Each point represents an IC and the colour is associated to its final state, the darker the tone of grey the shorter the cycle, fixed points are in black and divergent points in white. So, the different domain attractors (including the attractors) that coexist in the system can be seen here.

With the purpose of being able to distinguish the different coexisting areas, a diverse range of gray tones have been used on each figure. It must be taken into account that each figure has its own gray range, this means that, for example, an almost white area when $n_f = 5$ (Fig. 3.14.a) corresponds to a period of 6, while a darker area in a figure with higher n_f may correspond to a period higher than a thousand (Fig. 3.14.e). These figures allow reflecting the complex domains of attraction that appear when digitalizing.

It can be seen in Fig. 3.14 that the smaller the value of n_f the bigger the area of ICs that tends to diverge and to converge to fixed points. As n_f increases, the area of divergent and fixed points decreases. These figures along with Table 3.3 allows an easy interpretation of the system's behavior. In Table 3.3 the sequence lengths that appear in the attractor domain for every n_f are sorted by the more to the less numerous ICs that converges to that cycle. It can be seen the rate of occurrence. Indeed, figures with lower values of n_f present irregular, or rough surfaces, pointing out that different lengths cycles coexist there. For example, for $n_f = 5$ there is a prevalence of short periods cycles. In that case, there exist just two limit cycles, the lighter grey zone corresponds to the attraction domain of the limit cycles of length six, that is the less numerous cycle, according to Table 3.3, and, the darker zone corresponds to the attraction domain of length two cycle.

Although for $n_f \geq 13$ (Figures 3.14.i to 3.14.l) the attractor domain appears to be smooth and uniform, however, if we enlarge a section of the figures (Fig. 3.15) it can be seen that there are still cycles with different periods that coexist in the attractor for $n_f = 14, 17$ and 18 .

Nevertheless, when we want to make a general comparison of what happens to the periods when the precisions are varied a color scale is required, see Fig. 3.16.

Fig. 3.16 shows that as the value of n_f increases the colour of the area

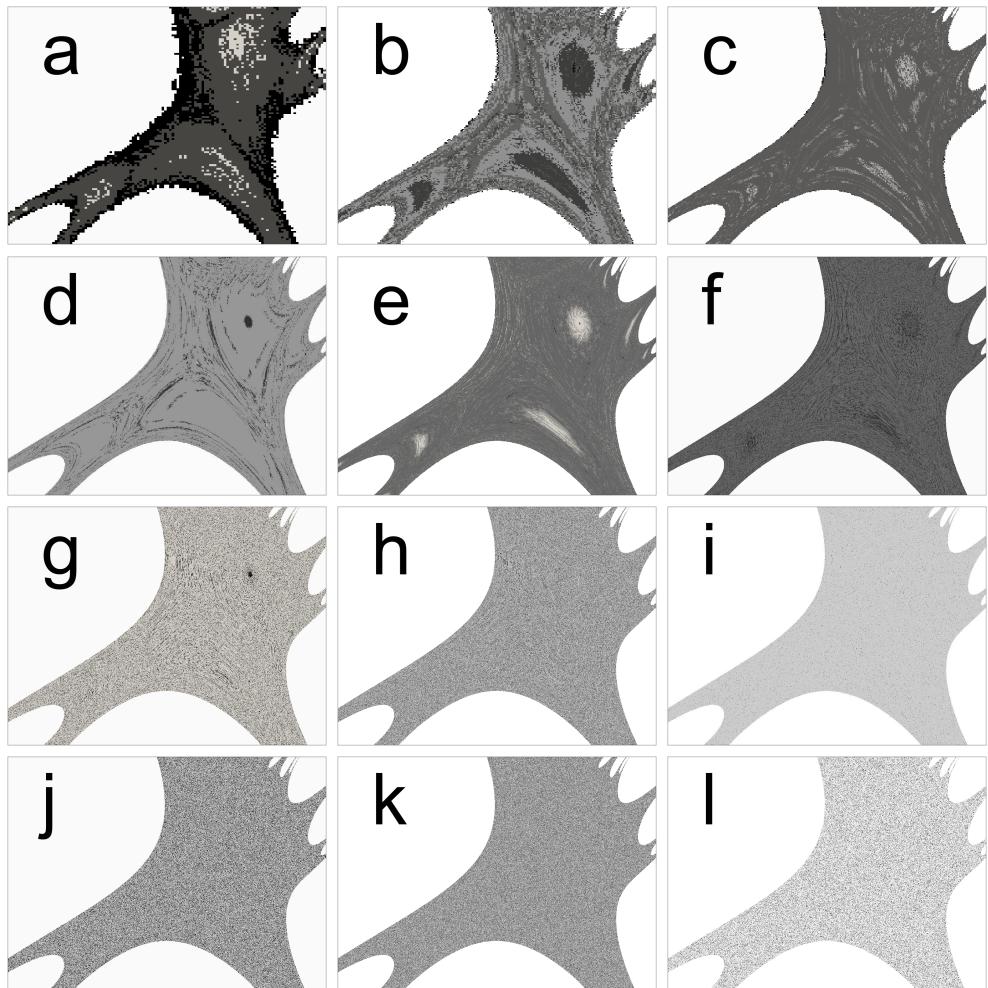


Figura 3.14: Coexisting areas in attraction domains for: (a) $n_f = 5$, (b) $n_f = 6$, (c) $n_f = 7$, (d) $n_f = 8$, (e) $n_f = 9$, (f) $n_f = 10$, (g) $n_f = 11$, (h) $n_f = 12$, (i) $n_f = 13$, (j) $n_f = 14$, (k) $n_f = 17$, (l) $n_f = 18$.

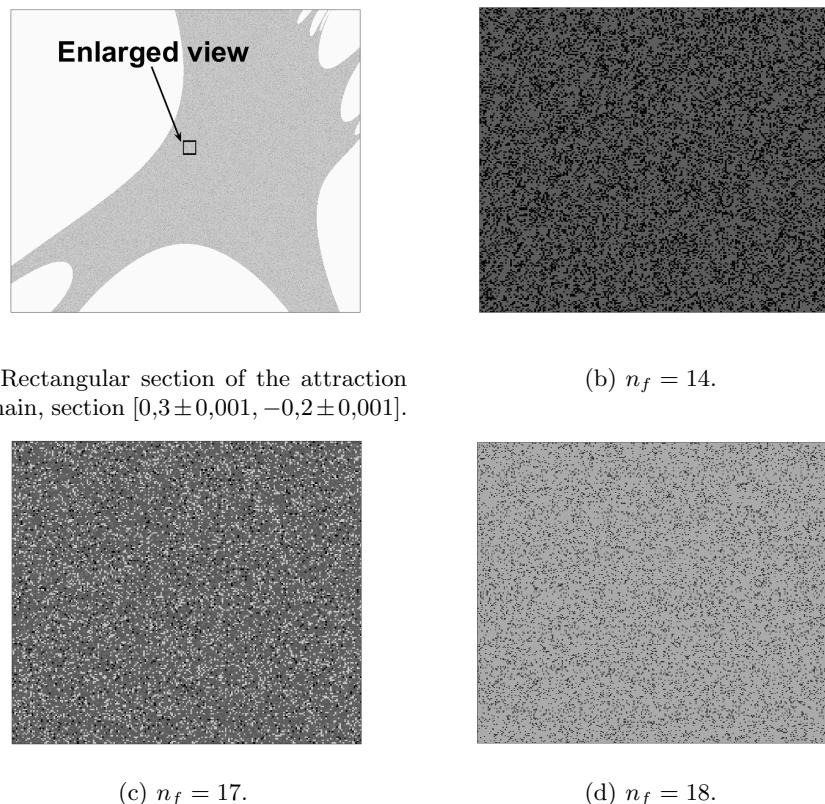


Figura 3.15: Enlarged views of sections of the attraction domains for higher values of n_f .

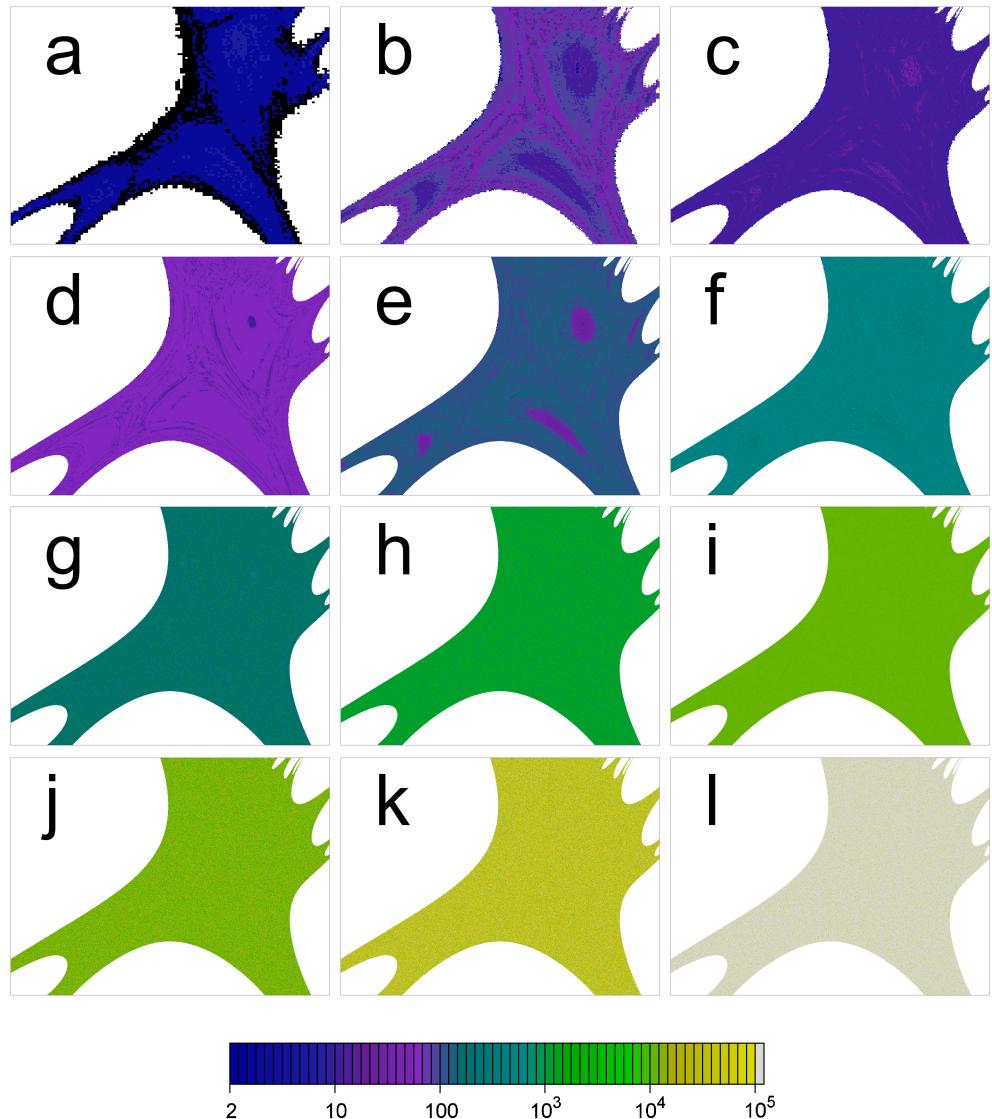


Figura 3.16: Period's lengths evolution of the attraction domains for: (a) $n_f = 5$, (b) $n_f = 6$, (c) $n_f = 7$, (d) $n_f = 8$, (e) $n_f = 9$, (f) $n_f = 10$, (g) $n_f = 11$, (h) $n_f = 12$, (i) $n_f = 13$, (j) $n_f = 14$, (k) $n_f = 17$, (l) $n_f = 18$.

Cuadro 3.3: Lengths of the periods within the attractor domain x and $y \in [-2, 2]$.

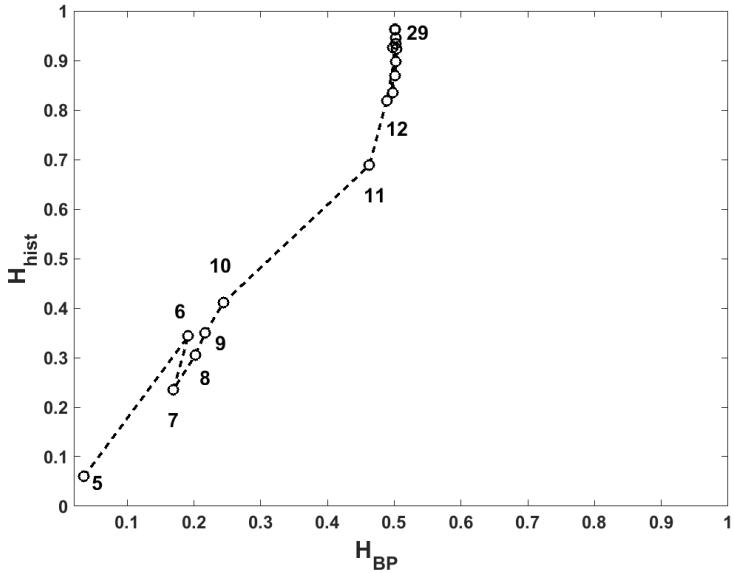
n_f	T (Percentage of ICs that converge to this period's length cycle)
5	2 (92,7 %);6 (7,3 %)
6	88 (41,6 %);44 (36,7 %);12 (13,8 %);16 (6,2 %);2 (0,8 %);24 (0,6 %);26 (0,2 %)
7	12 (83,5 %);14 (8,9 %);24 (5,2 %);34 (1,8 %);2 (0,6 %)
8	68 (91,7 %);14 (6,2 %);12 (1,8 %);17 (0,2 %);15 (0,1 %)
9	140 (54,5 %);123 (25,4 %);34 (8,6 %);44 (4,3 %);38 (3,9 %);22 (2,9 %);48;2;12;4 (< 0,1 %)
10	655 (78,2 %);212 (21,1 %);143 (0,5 %);12 (0,1 %);2;36;13;20;10;4 (< 0,1 %)
11	153 (78,1 %);461 (10,8 %);1381 (8,7 %);434 (2,3 %);18;30;53;32;34;10;2 (< 0,1 %)
12	2,278 (64,4 %);438 (22,4 %);598 (7,6 %);886 (4,7 %);12 (0,7 %);87;2;42;23;32;10 (< 0,1 %)
13	11,510 (98,9 %);1052 (1 %);12;26;2;10 (< 0,1 %)
14	21,333 (69,2 %);5,804 (16,5 %);4,795 (7,9 %);1,264 (5,8 %);2,429 (0,5 %) 46;23;21;10;12;17 (< 0,1 %)
15	10,099 (58,6 %);1,762 (19,4 %);14,887 (18,3 %);1,598 (3,4 %);750;105;23;14;2;10 (< 0,1 %)
16	54,718 (87,5 %);5,017 (4,7 %);> 10^5 (3,7 %);5,367 (2,5 %);703 (0,9 %) 1,159;1,802 (0,2 %);377;75;10 (< 0,1 %)
17	37,812 (53,1 %);38,456 (24,1 %);> 10^5 (16,0 %);34,749 (3,0 %);3,362;718 (1,5 %) 3,006,5,222 (0,1 %);15 (< 0,1 %)
18	> 10^5 (87,4 %);52,069 (12,5 %);2,471 (0,1 %);146;51 (< 0,1 %)
<i>float</i>	> 10^5 (100 %)

becomes more smooth and clear, indicating that the ICs converge to higher periods cycles. This is the range of initial values that generate useful sequences increases for higher values of n_f .

This can also be seen in Table 3.3, where as n_f increases the predominant limit cycle's length increases. In order to compare the obtained values with the real sequences we have simulated in floating-point with 236-bit mantissa (IEEE 754 octuple-precision binary floating-point format) we call this here *floating-point* or just *float*, it is the arithmetic closest to real numbers. Then, using float precision all the limit cycles are higher than 10^5 , they converge to the chaotic attractor seen in Fig. ??d.

In relation to the randomness quantifiers, we realized that the analysis performed up to this point was not enough to fully describe the changes in the dynamic of a digitalized chaotic system. To reach long periods does not ensure that the systems' exhibit good properties with respect to randomness. So we decided to further study the data obtained by employing statistical quantifiers.

As said, in Fig. 3.14.a the two gray zones correspond to the initial conditions that converge to the two coexisting cycles of period two and six respectively. Then this two cycles will have a determined value of H_{hist} and H_{BP} , $H_{hist} |_{T=2} =$

Figura 3.17: Plane H_{hist} - H_{BP} for different number of bits.

0,0625, $H_{hist}|_{T=6} = 0,1199$, $H_{BP}|_{T=2} = 0,1053$ and $H_{BP}|_{T=6} = 0,2723$. However, the reported value of these quantifiers can not be the average of both, since the rate of occurrence of cycle two is much greater than that of cycle six (period two appears 92,7 % times while period six only 7,3 %, see Table 3.3). Therefore, we have calculated the averaged quantifiers by weighting each quantifier by its rate of occurrence.

The H_{hist} vs H_{BP} plane, shown in Fig. 3.17, allows a quick visualization of the behavior in terms of randomness of the system, in this plane the “ideal” point, from the statistical point of view, is (1, 1). Here, the system seems to stabilize for n_f higher than 12. It can be seen that while the H_{hist} stabilizes close to the maximum value ($H_{hist} = 1$), the H_{BP} tends to stabilize to 0,5. This value of H_{BP} is characteristic of chaotic systems and is due to the inner structures of their attractors.

A summary of the observed analysis of these outputs can be seen in Fig. 3.18.

Fig. 3.18.a and 3.18.b show the number of points that diverge and converge to fixed points respectively as the value of n_f increases, in both cases, the final value tends to the floating-point case. It is clear from these figures that for $n_f \sim 12$ the system seems to have stabilized. Figure 3.18.c shows that the

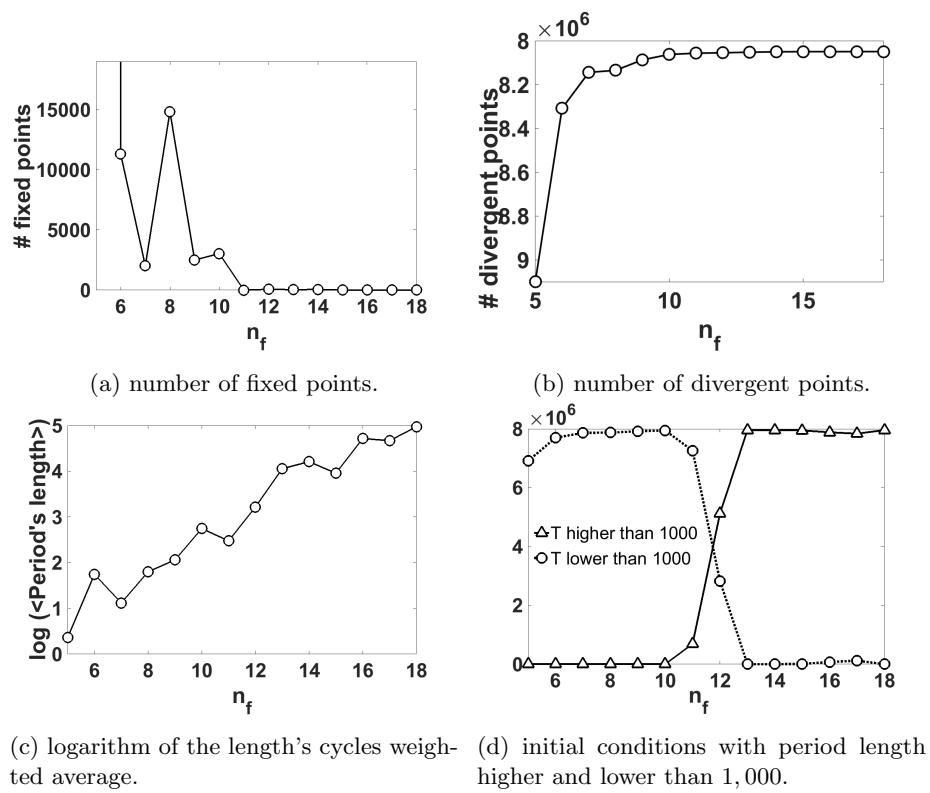


Figura 3.18: Summary of initial conditions' behavior.

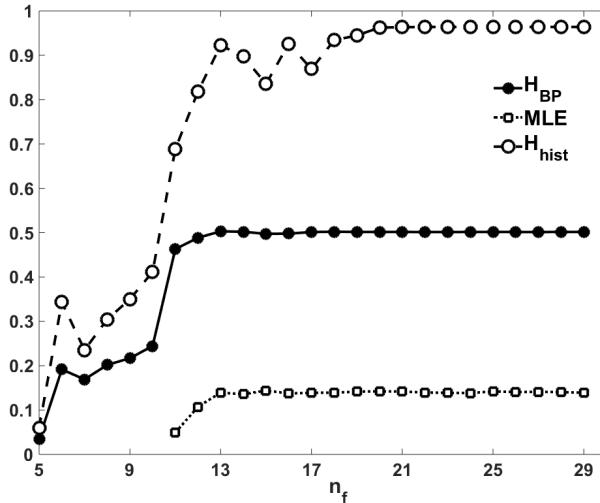


Figura 3.19: Weighted average of quantifiers H_{BP} , H_{hist} and MLE as functions of the number of bits.

averaged period of cycles increases at a logarithmic rate. Finally, Fig. 3.18.d shows the number of initial conditions that present periods T higher and lower than 1,000. Again, a value of 12 for n_f seems to be the limit to obtain a good approximation of the system.

Figure 3.19 shows the weighted average of quantifiers H_{hist} , H_{BP} and MLE . In the figure it can be seen that the three quantifiers tend to the value calculated using floating-point arithmetic. While H_{BP} and MLE stabilize for $n_f \sim 12$ or 13, H_{hist} reaches the floating-point value for $n_f \sim 19$, showing that there are properties of the output sequences that only this quantifier can detect. This confirms the need to use both quantifiers to characterize the randomness of the sequences.

As can be seen from the above analysis, the minimum number of bits is determined by H_{hist} quantifier and results to be $n_f = 19$ plus the number of bits used to represent the integer part $n_i = 4$, therefore n_{min} turns out to be equal to 23.

3.6.4. Conclusion

In this work, we have developed a detailed analysis of the changes in behaviour of a 2D-quadratic map fixed-point implementation. Our goal is to report

the rate of degradation for each systems' property, so as to be used by authors at the time of designing their particular applications. Results show that it is possible to determine a threshold for the number of bits employed in the fixed point representation of the system, whereas the domain of attraction preserves its integrity and the characteristics of the generated sequences are kept. With the help of the quantifiers of randomness introduced it was possible to determine that limit, in the case of study it was 23 bits. The same procedure should be repeated for any other system if it is desired to be used in a digital electronic application, such as controlled noise generators or to develop novel encryption systems. In the particular case of Sprott's chaotic system, if the minimum number of 23 bits is satisfied at the time of digitalizing, we conclude that it is possible to successfully use it as a component of new encryption algorithms.

Capítulo 4

Generadores de TRNG usando ROs en FPGA

4.1. Introduction

El *Jitter* es cualquier desviación leve del período medio de una señal presumtamente periódica. Hay muchos ejemplos físicos donde la inestabilidad es relevante. Algunos ejemplos de diferentes áreas son: (a) Stalberg *et. al* [?] encontraron que el intervalo de tiempo entre los dos potenciales de acción de las fibras de dos fibras musculares, que pertenecen a la misma unidad motora en los músculos humanos normales, muestra una variabilidad o inestabilidad; (b) Mecozzi *et. al* [?] detectaron jitter de temporal y variaciones de amplitud en enlaces ópticos utilizando transmisión de pulso altamente disperso; (c) Derickson *et. al* [?] realizó una comparación completa de la fluctuación de tiempo en el caso de los láseres semiconductores en modo bloqueado; (d) el California y Carnegie Planet Search en el Observatorio Keck [?] informó la inestabilidad de las estrellas en las velocidades radiales; (e) Roberts & Guillemin estudiaron los retardos debidos a las colas en etapas de *upstream multiplexing*, en una red de Modo de transferencia asíncrono (ATM); (f) Baron et al [?] consideró la calidad de la señal del *bunch clock* del *Large Hadron Collider* (LHC), en términos de inestabilidad, un problema fundamental porque sincroniza todos los sistemas electrónicos en el detector; (g) Marsalek *et. al* analizaron la relación entre la en-

trada sináptica y la fluctuación de fase de salida pico en neuronas individuales [?], etc.

Además, los instrumentos digitales se utilizan en cualquier experimento moderno y la inestabilidad inevitable en los sistemas de adquisición de datos produce incertidumbres en el tiempo y, por consiguiente, en cualquier determinación del espectro.

Este capítulo está dedicado a los osciladores de anillos (*RO*). Recalcemos que en esta aplicación particular, el *jitter* no siempre es indeseable. El *jitter* no es deseado en aplicaciones que usan un *RO* como generador de reloj [?, ?, ?, ?, ?]. Por el contrario, los generadores de números aleatorios *RNG* basados en *RO*'s, usan *jitter* como fuente de aleatoriedad, [?, ?]. El *jitter* también mejora la compatibilidad electromagnética para distribuir la frecuencia del reloj sobre una banda, mejorando la Compatibilidad Electromagnética (EMC) [?].

La determinación del *jitter* de fase en *RO* se ha estudiado en varios artículos: en [?] se presentó el estudio de tres medidas relevantes del *jitter* en el dominio del tiempo. En [?] se propuso un modelo para la generación y distribución del *jitter* en *RO*. En este artículo, los autores separan las fuentes de inestabilidad en deterministas y aleatorias (gaussianas); además, cada fuente se clasifica adicionalmente en local o global. Demuestran que las contribuciones más importantes son la inestabilidad gaussiana local y la inestabilidad determinística global y solo la primera debe usarse como una fuente de aleatoriedad de generadores de números aleatorios verdaderos (*TRNG*). El mismo enfoque se usó en [?, ?, ?, ?]. En Lubicz *et. al* se describe un método práctico y eficiente para estimar la tasa de entropía de un *TRNG* basado en osciladores libres; enfatizaron que su método no requiere extraer las señales del dispositivo y analizarlas con equipos externos [?] (una metodología que introduce fluctuación y distorsión extra en la señal medida debido a la cadena de adquisición de datos).

Por lo general, *jitter determinista* es el nombre que se le da a cualquier *jitter* *no gaussiano*. Está limitado y se caracteriza por su valor máximo de Δ_{pp} . *Jitter aleatori* es el nombre utilizado para la *jitter* gaussiano y se caracteriza por su valor RMS. A veces aparece *jitter* periódico determinístico. Tiene un *periodo* que es el intervalo entre dos tiempos el efecto máximo (mínimo); el inverso del período de tiempo es *la frecuencia del jitter*. El *jitter* periódico con una

frecuencia de fluctuación inferior a $10Hz$ usualmente se denomina *wander* y el nombre *jitter* está reservada solo a la fluctuación periódica con frecuencias en o por encima de $10Hz$. En comunicaciones, *jitter total* es $T = \Delta_{pp} + 2nR_{rms}$ donde n es un número entre 6 y 8 relacionado con la tasa de error de bit (*BER*).

Los *ROs* son uno de los principales componentes de los circuitos integrados analógicos y digitales y se han utilizado ampliamente como osciladores *on-chip* para generar relojes en circuitos de alta velocidad. Además, los *ROs* se pueden implementar fácilmente en circuitos digitales programables como *FPGAs*. Las principales ventajas de los osciladores integrados *RO* sobre los *LC* son su área de chip más pequeña, su rango de funcionamiento más amplio (que puede ser sintonizado eléctricamente) y su menor consumo de energía.

Ya sea que se quiera usar o eliminarlo, el *jitter* en *ROs* debe medirse, lo que no es una tarea simple. La principal contribución de este trabajo es proporcionar una técnica de medición del *jitter* basada en cuantificadores de la teoría de la información (*ITQ*). Utilizamos un modelo estocástico cuya aleatoriedad está relacionada con la amplitud de la inestabilidad. Cada *ITQ* propuesto utilizado en este trabajo se basa en una entropía, es decir, una función de Shannon de la función de distribución de probabilidad (*PDF*) asignada a la serie de tiempo del proceso estocástico. También se pueden usar desequilibrios y complejidades [?, ?], pero no representan una mejora en nuestro caso. En trabajos anteriores [?, ?] mostramos que muchas *PDFs* diferentes pueden asignarse a la misma cadena de datos. La mejor opción depende de la aplicación específica. En este caso utilizamos dos opciones para *PDF*: el *histograma normalizado* y el *histograma de patrones de orden*. Se usa un plano de representación para comparar diferentes situaciones. Una vez que se elige la *PDF*, la Entropía de Shannon es la función básica que cuantifica la uniformidad de la *PDF*. Las *entropías normalizadas*, *entropías diferenciales* y *tasa entropía* son las otras *ITQs* evaluadas. En nuestro caso las *entropías diferenciales* obtienen los mejores resultados y se utiliza un *plano de entropías diferenciales* para comparar su sensibilidad como medida de *jitter*.

4.2. Determinación del *jitter* en *RO*'s

Hay dos situaciones diferentes en lo que concierne al *jitter* en *ROs*: (a) en algunas aplicaciones es suficiente con asegurar que el *jitter* no perturba a la señal por encima de un límite aceptable. En este caso la señal se observa en un osciloscopio con un amáscara sobre la pantalla, lo que es suficiente para verificar que la señal se mantiene dentro de los márgenes de tolerancia; (b) en otros casos se precisa una determinación exacta del *jitter*. Entre esos casos está la caracterización de *ROs* considerada en este trabajo.

Los *ROs* ideales están compuestos por un numero impar de inversores. Cada inversor tiene un tiempo de propagación y por lo tanto los flancos de subida y bajada separados por medio período viajan a través de los inversores. Si todos los tiempos de propagación son constantes, la salida de este *RO* ideal es una señal cuadrada con un espectro de frecuencia discreto. Pero los tiempos de propagación no son constantes, por lo tanto hay *jitter*. El *jitter* distorsiona el espectro de potencia ensanchando cada delta en un máximo con cierta anchura.

Supongamos que $T/2$ es el medio período de un *RO* ideal. Entonces está dado por:

$$\frac{T}{2} = k \sum_{i=1}^k d_i \quad (4.1)$$

En donde k es el número de inversores y d_i es el tiempo de propagación a través del i -ésimo inversor. Cuando hay jitter, d_i es una variable aleatoria que modelamos como:

$$d_i = D_i + \Delta d_i \quad (4.2)$$

donde D_i es el valor medio de d_i con el nivel nominal de voltaje de fuente y la temperatura normal, y Δd_i es la variación del retardo producida por los eventos físicos locales y los cambios globales en las condiciones de trabajo del dispositivo (como V_{CC} , temperatura , etc.). Entonces, el *jitter* en *ROs* se evidencia por el desplazamiento aleatorio de la ubicación de los flancos ascendentes (descendentes), con respecto a la ubicación perfectamente periódica. La medición directa de este desplazamiento tiene dos problemas principales: (a) requiere un instru-

mento de muy alta frecuencia, porque la resolución del tiempo está limitada por el período de muestreo T_s ; (b) esta técnica introduce fluctuaciones y distorsiones adicionales en la señal medida proveniente de la cadena de adquisición de datos. Entonces es más conveniente usar *medidas indirectas*, por medio de variables aleatorias auxiliares relacionadas con las propiedades estadísticas relacionadas con el *jitter* para medir la fluctuación de fase con una perturbación mínima [?].

El procedimiento general es el siguiente:

1. Se muestrea la salida con el período de muestreo T_s para obtener una serie de tiempo binaria. En el caso ideal de *no-jitter*, la salida es una *onda cuadrada continua y perfectamente periódica* con un período T . Entonces es posible ajustar T_s para hacer $T/2 = mT_s$ con $m \in N^+$. La serie de tiempo binaria será periódica con m unos seguida de m ceros. Cuando el *jitter* está presente, la serie binaria no es periódica sino estocástica. Este modelo estocástico se conoce como *proceso de renovación alterna*.
2. Se pueden usar muchos cuantificadores de aleatoriedad diferentes para caracterizar el modelo estocástico asociado con el *jitter* medido. En este trabajo utilizamos cuantificadores de la teoría de la información.

Tengamos en cuenta que el *jitter* es acumulativo y surgen dos situaciones básicas: (a) si se supone que el *jitter* introducido por cada etapa es totalmente independiente del *jitter* introducido por otras etapas, significa que $\sigma_T^2 = m * \sigma_s^2$, en donde σ_s es el *jitter* de cada muestra, y se supone que todas las muestras tienen fluctuaciones con la misma distribución normal; (b) si las fuentes de *jitter* están totalmente correlacionadas entre sí, entonces $\sigma_T = m * \sigma_s$.

4.3. Resultados

Se simuló con Matlab® una salida de un *RO* muestreada uniformemente sin *jitter* y se generó un archivo de salida con una longitud de $N_b = 7,000,000$ de bits. Se exploró un conjunto de cien valores de relación de muestreo $r = T_s/T \in [6, 5; 9, 5]$ (donde T_s es el período de muestreo y T es el período de salida *RO*). *Jitter* con una distribución normal con un conjunto de diferentes valores de varianza σ_s (ver a continuación) se agregaron y se generaron nuevos archivos de

la misma longitud. Nuestro método emula el verdadero proceso de muestreo de la salida ruidosa de un *RO* real; el código detallado se publicó en Mathworks [?].

Por cada valor de σ_s , ten surrogates (each one with a different random initial condition) were generated and new files with N_b bits each were stored. It was assumed that jitter of individual samples is independent, normal distributed random variables, with zero mean value and variance $\sigma_i = \sigma_s$. Consequently, the variance of the accumulated jitter over one period T is given by $\sigma_T^2 = r\sigma_s^2$ [?]. The values considered are $\sigma_T = \{0, 0,001, 0,002, 0,003, 0,004, 0,005, 0,007, 0,01, 0,02, 0,02, 0,04, 0,05, 0,07, 0,1\}$.

Para cada valor de σ_s , se generaron diez surrogados (cada uno con una condición inicial aleatoria diferente) y se almacenaron nuevos archivos con N_b bits cada uno. Se asumió que el *jitter* de las muestras individuales es independiente, con variables aleatorias distribuidas normales y un valor medio cero y varianza $\sigma_i = \sigma_s$. En consecuencia, la varianza del *jitter* acumulada durante un período de T está dada por $\sigma_T^2 = r\sigma_s^2$ [?]. Los valores considerados son $\sigma_T = \{0, 0,001, 0,002, 0,003, 0,004, 0,005, 0,007, 0,01, 0,02, 0,02, 0,04, 0,05, 0,07, 0,1\}$.

Por cada archivo se evaluaron todos los cuantificadores definidos en la sección ?? para $D \in [2, 10]$ y $W \in [1, 26]$.. Los detalles sobre la evaluación, las ventajas y los inconvenientes de cada cuantificador se informan en la sección ??: ellos son S_W , $S_{BP}^{(D)}$, H_W , $H_{BP}^{(D)}$, h and h^* . Aquí mostraremos solo los resultados más relevantes para mostrar la razón por la cual los dos últimos cuantificadores (h y h^*) resultan los más adecuados para este problema.

- En el caso de entropía normalizada H_W , depende en gran medida de W . Además, el análisis de H_W en función de r muestra que no permite determinar un valor óptimo de la relación de muestreo r (ver Fig. 4.1). Este es un problema importante si los cuantificadores se van a usar para configuraciones experimentales.
- En el caso de la entropía de Bandt & Pompe normalizada $H_{BP}^{(D)}$, también está presente una fuerte dependencia con la dimensión de embedding D . Nuevamente, no es fácil determinar el valor óptimo de r del análisis de este parámetro en función de r (ver Fig. 4.2).

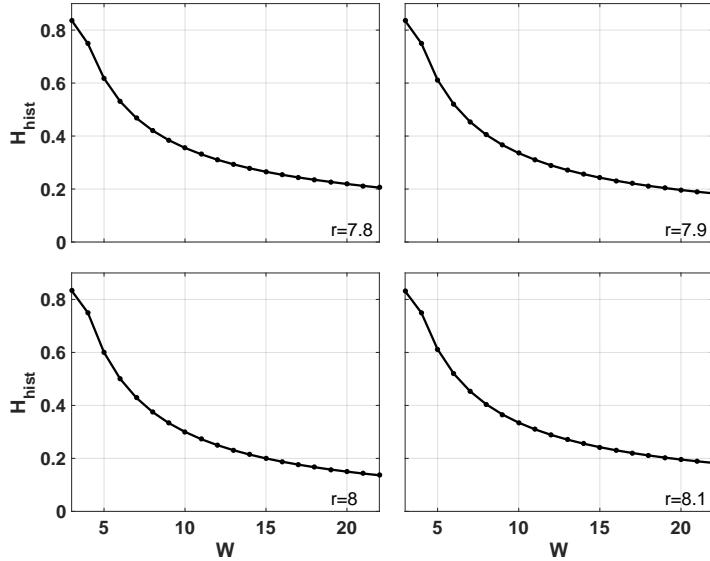


Figura 4.1: Entropía normalizada H_W en función de W para un *RO* sin *jitter* muestreado con diferentes valores de r .

- Un comportamiento similar aparece en todos los otros funcionales relacionados con estas dos entropías. En resumen, nuestros resultados muestran que tanto h y h^* son independientes de cualquier parámetro arbitrario utilizado en su determinación estadística. Estos dos cuantificadores también se han considerado en dos excelentes artículos [?, ?].

Estos resultados muestran que los dos cuantificadores, h y h^* , son apropiados para ser usados como medidores de *jitter* debido a que:

- (a) Para $\sigma_T = 0$ (salida sin *jitter*) se acercan rápidamente a un valor límite constante ya que tanto D como W tienden a ∞ y este valor es independiente de D y W ;
- (b) Son funciones monótonas y proporcionales de σ_T .
- (c) A partir de su análisis, es posible detectar el valor óptimo de la relación de muestreo r . En las siguientes figuras mostraremos estas afirmaciones que son representativas de todos nuestros resultados.

La figura 4.3 muestra la entropía diferencial de Bandt & Pompe h^* , como función de D , con W como parámetro, para un *RO* sin *jitter*. Se puede ver que existe un

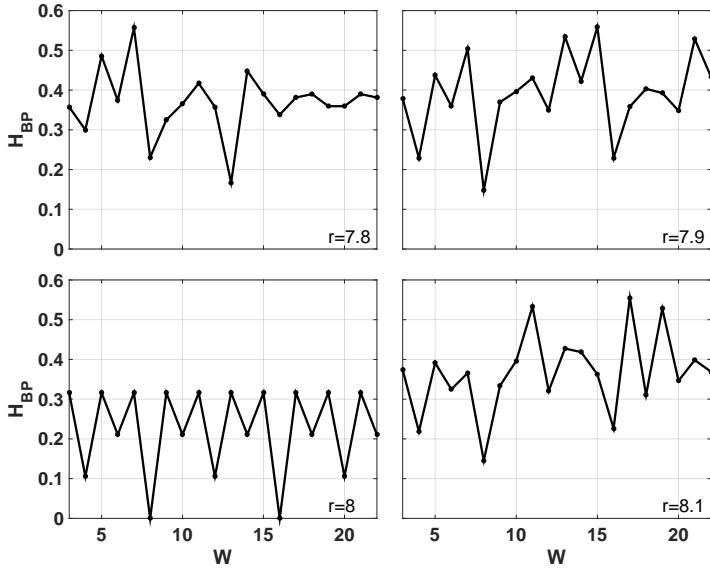


Figura 4.2: $H_{BP}^{(D)}$ en función de W para un *RO* sin *jitter* muestreado con diferentes valores de r . Los cálculos fueron hechos con superposición de palabras

valor umbral $W = 4$ sobre el cual todas las curvas colapsan en una sin importar el valor de D . Además, la Fig. 4.3 también muestra que para $D \geq 8$ todas las curvas colapsan en una, independientemente del valor de W . En conclusión, si $D \geq 8$ y $W \geq 4$ obtenemos un cuantificador independiente de D y W . La influencia del *jitter* en este cuantificador se muestra en la figura ??, donde h^* se representa como una función de D con σ_T como parámetro. Los valores considerados son $\sigma_T = \{0(\sin jitter), 0,001, 0,002, 0,003, 0,004, 0,005, 0,007, 0,01, 0,02, 0,02, 0,04, 0,05, 0,07, 0,1\}$. El recuadro de la Fig. 4.4 muestra h^* como una función de σ_T para $D = 8$. Este recuadro muestra que este cuantificador es una función monótona creciente de σ_T . Finalmente, la Fig. ?? muestra h^* como una función de la relación de muestreo r . En esta figura, se muestra que hay un mínimo para el r correcto (en este caso $r = 8$). Además, la sensibilidad de h^* en función del *jitter* es máxima para este mismo valor ideal de r .

Analicemos ahora el segundo cuantificador, h . Este cuantificador solo depende de W porque D no se usa para definir la *PDF* asignada a la serie de datos. La Fig. ?? muestra un caso sin *jitter*, h es independiente de W para $W \geq 4$. Para lo siguiente adoptamos $W = 6$. La figura ?? muestra la influencia del *jitter* sobre este cuantificador. Queda claro en el recuadro de esta figura que, para el

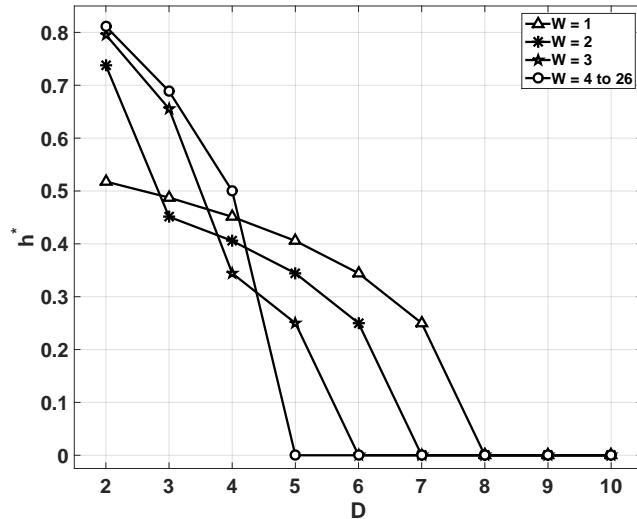


Figura 4.3: h^* en función de D para un RO sin *jitter* muestreado con $r = 8$.

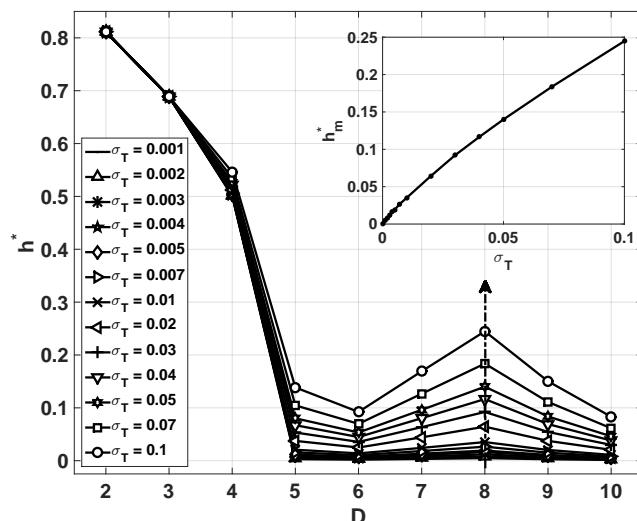


Figura 4.4: h^* en función de D para un RO muestreado con $r = 8$ con longitud de palabra $W = 6$ para *jitter* con diferentes varianzas. El recuadro muestra h^* en función de σ_T para $r = 8$, $W = 6$ y $D = 8$.

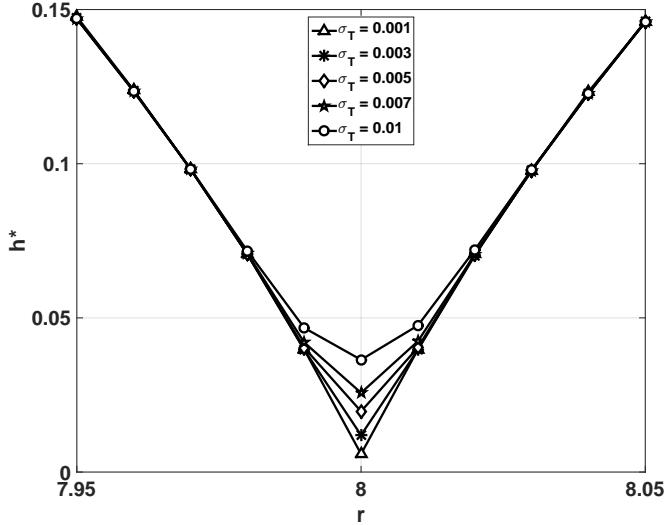


Figura 4.5: h^* en función de r para $r \in [7.95, 8.05]$, con algunos σ_T , $W = 6$ y $D = 8$. La curva tiene un mínimo en el valor correcto de $r = 8$.

valor seleccionado $W = 6$, h es una función monótona creciente de la varianza del jitter σ_T . La Fig. ?? muestra que h tiene un mínimo cuando r toma su valor óptimo ($r = 8$). Note que este mínimo es robusto también en presencia de jitter.

Se debe realizar un análisis adicional para asegurar que los valores seleccionados $W = 6$ y $D = 8$ produzcan archivos simbólicos con una buena estadística. Para un alfabeto dado \mathcal{A} con m elementos, y un archivo simbólico dado de longitud n , el parámetro de calidad $\alpha = n/m$, vea ???. La calidad es mejor a medida que α aumenta y se acepta un valor mínimo $\alpha = 10$. De acuerdo con la sección ?? los valores seleccionados $W = 6$ y $D = 8$ proporcionan $\alpha_h \simeq 10^5$, $\alpha_{h^*} \simeq 175$ con superposición y 29 sin superposición. Todos los casos dan $\alpha > 10$ como es requerido.

Figure 4.9 shows the $h_m^* \times h$ plane. The quantifiers have been calculated sweeping the values of D from 2 to 11, and W from 2 to 26 (both for h^* and only W in the case of h_{hist}). Better differentiation is obtained for higher values of the parameters, this is because both quantifiers tend to quantify the source value for D and W equal to infinite. Nevertheless, this is impossible in real practice, also the amount of data available limits the values of the parameters based to achieve good statistics. Therefore, we seek the minimum values (threshold value)

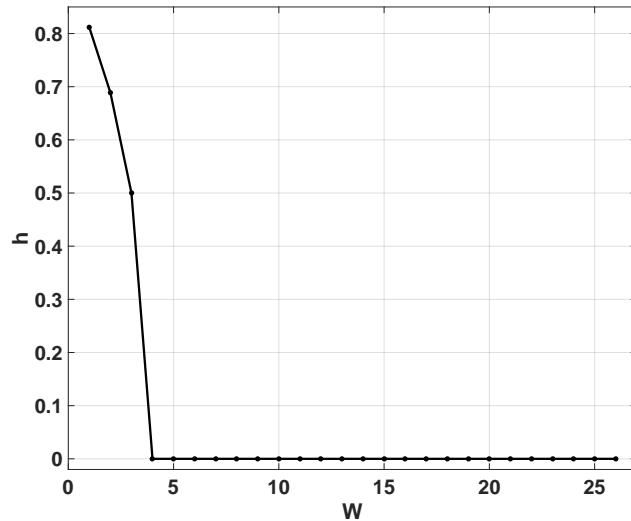


Figura 4.6: h en función de W para un RO sin *jitter* muestreado con $r = 8$.

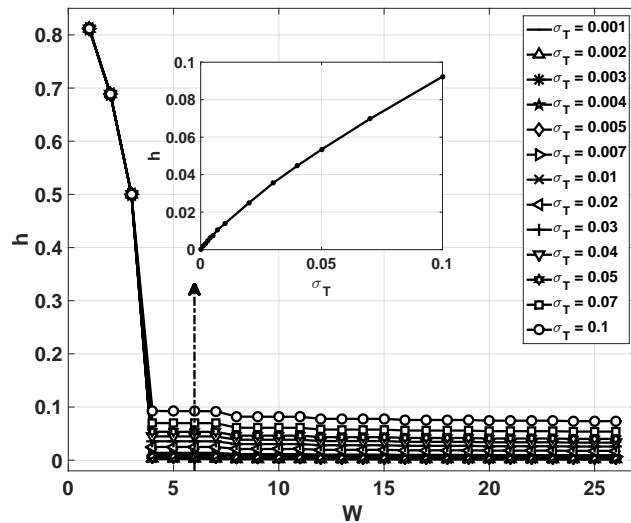


Figura 4.7: h en función de W para un RO muestreado con $r = 8$, con *jitter* con distintas varianzas. El recuadro muestra h en función de σ_T con $r = 8$ y $W = 6$.

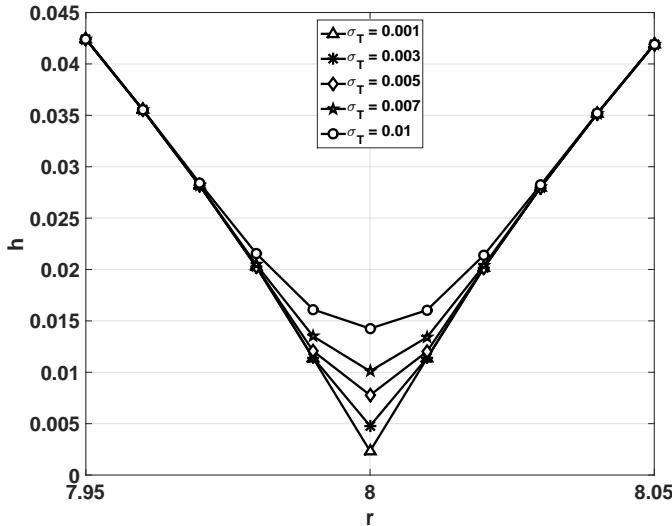


Figura 4.8: h en función de r con $r \in [7,95,8,05]$, para distintos σ_T y $W = 6$. La curva tiene un mínimo en $r = 8$.

of the parameters that discern the jitter good enough. In Figure ?? it can be seen that for values of W equal to 3 or lower the h_{hist} quantifier is insensitive to the variations of the jitter. Therefore, W should be 4 or higher, this result is in concordance with the threshold determined in Figures ???. In the case of h^* the W value should be equal or higher than 4 and the value of D higher than 7, again this result is in concordance with the ones derived from Figures ?? and ???. The bottom left area of the plane is the one with better performance of both quantifiers.

La figura ?? muestra el plano $h_m^* \times h$. Los cuantificadores se calcularon barriendo los valores de D de 2 a 11 y W de 2 a 26 (barrimos ambos para h^* y solo W en el caso de h_{hist}). Se obtiene una mejor diferenciación para valores más altos de los parámetros, esto es porque ambos cuantificadores tienden a cuantificar la entropía de la fuente cuando D y W tienden a infinito. Sin embargo esto es imposible en la práctica real, la cantidad de datos disponibles limita los valores de los parámetros para lograr buenas estadísticas. Por lo tanto, buscamos los valores mínimos (valor umbral) de los parámetros que distinguen el *jitter* lo suficientemente bien. En la figura ?? se puede ver que para valores de W iguales o menores a 3, el cuantificador h_{hist} no es sensible a las variaciones del

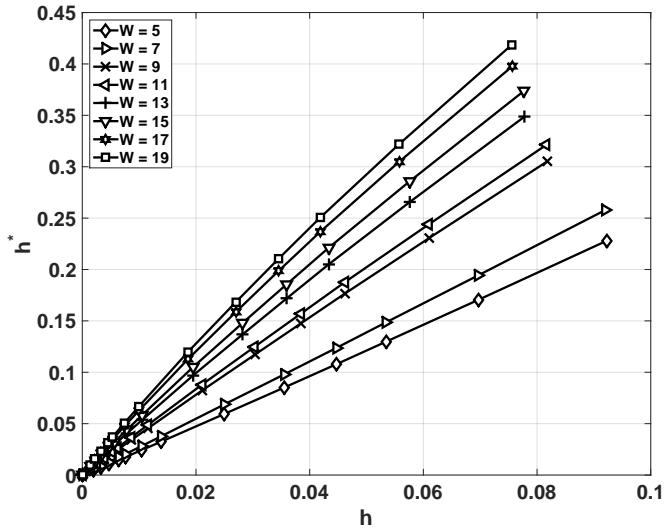


Figura 4.9: h^* as a function of h for $r = 8$, $D = 8$ and different values of W .

jitter. Por lo tanto, W debe ser o superior, este resultado está en concordancia con el umbral determinado en las Figuras ???. En el caso de h^* el valor de W debe ser igual o superior a 4 y el valor de D superior a 7, nuevamente este resultado está en concordancia con los derivados de las Figuras ?? y ???. El área inferior izquierda del plano es la que tiene un mejor rendimiento de ambos cuantificadores, sin embargo es la que precisa un mayor esfuerzo de cómputo y una mejor estadística.

Una comparación entre ambos cuantificadores se muestra en la figura 4.9. Los marcadores corresponden a varianzas $\sigma_T = \{0, 0,001, 0,002, 0,003, 0,004, 0,005, 0,007, 0,01, 0,02, 0,03, 0,04, 0,05, 0,07, 0,1\}$. Hay que tener en cuenta que la pendiente de cualquiera de estas curvas es dh^*/dh y es igual al cociente entre pendientes de curvas en las inserciones de las Figs. ??, y ???. Si $dh^*/dh \rightarrow 1$, h^* es más sensible que h para medir el *jitter*. La pendiente aumenta levemente de $\sim 2,47$ para $W = 5$ a $\sim 5,54$ para $W = 19$, esto muestra que h^* se vuelve más sensible a medida que aumenta W .

También evaluamos h^* sin la superposición de bits entre números naturales consecutivos pero manteniendo la superposición de los $D - 1$ números naturales entre patrones de orden (en todos los casos h se evaluó con la superposición de $W - 1$ bits consecutivos). Los resultados se representan en la Fig. ?? donde se

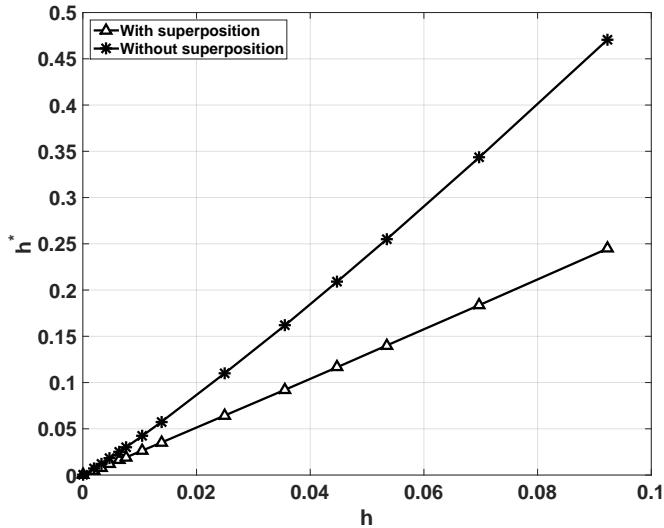


Figura 4.10: h^* as a function of h for $r = 8$, $W = 6$ and $D = 8$. Two procedures to obtain W -bits natural numbers are considered: with and without superposition (see text).

muestra que al eliminar la superposición aumenta la sensibilidad de este cuantificador. Por supuesto, obtenemos una cantidad menor de W bits en números naturales del archivo original de siete millones de binarios, y en consecuencia, la calidad estadística es menor que la del cálculo original con superposición. Para aumentar α hasta su valor anterior, se requieren archivos binarios más largos.

4.4. Conclusions

Dada su utilidad como *PRNG* y generadores de reloj, los *ROs* se están convirtiendo en uno de los principales componentes básicos de los circuitos digitales. El jitter es inevitable en *ROs*, y en consecuencia, necesita ser caracterizado. La mezcla y la distribución de valores son las principales propiedades a considerar. Varios *ITQ* fueron evaluados aquí. S_W , $S_{BP}^{(D)}$, H_W y $H_{BP}^{(D)}$ resultan dependientes de los parámetros W y D . Esto es un inconveniente si los usamos como medidas de inestabilidad. Por otro lado, no es posible calcular *rate entropies*, h_0^* y h_0 , ya que se necesita una cantidad infinita de datos para su cálculo. Las dos *entropías diferenciales*, h^* y h , en cambio, son independientes de los parámetros utilizados para su determinación y son estimadores de la *rate entropy*. Hemos mostrado

4.5. IMPLEMENTACIÓN Y ANÁLISIS ESTADÍSTICO DE TRNG BASADO EN ROS109

en la sección ?? que en el caso de *ROs* muestreados, presentan un mínimo para la tasa de muestreo correcta, lo que los convierte en una buena medida de la calidad tanto de los *ROs* como de los *PRNGs* derivados de ellos.

El plano de entropía dual determinado por estos cuantificadores ha demostrado discernir satisfactoriamente entre las dos principales propiedades deseadas de *PRNG*, la equiprobabilidad entre todos los valores posibles y la independencia estadística entre valores consecutivos. Por lo tanto, permite ver claramente lo que debe mejorarse en una secuencia determinada. Los ejemplos presentados aquí han demostrado la necesidad de utilizar ambos histogramas para caracterizar secuencias.

4.5. Implementación y análisis estadístico de *TRNG* basado en *ROs*

4.5.1. Resumen

This paper deals with the use of Ring Oscillators (*ROs*) as pseudo random number generators (*PRNG*). The design, made for *ALTERA Cyclone III* ©, using low level primitives is explained. Two relevant characteristics of a *PRNG* are considered to validate the design: 1) the equiprobability of all possible outcomes and 2) the statistical independence of consecutive values. In this work these properties are measured via Information Theory Quantifiers. A dual entropy plane is used to represent the time series and easily visualize the results obtained with different configurations. The quality is also compared with other available *PRNGs* by means of the dual entropy plane. Our method constitutes an effective reduction of the complete analysis made with test suites like *DIEHARD* or *NIST*.

4.5.2. Introducción

The jitter and phase noises present in ring oscillators, are not convenient in several applications of *ROs*, for example in the implementation of *on-chip oscillators* to generate clocks in high-speed circuits[?, ?, ?]. However they are the source of randomness for *RO-based PRNG* [?, ?]. Furthermore *ROs* can

be implemented in a full-digital circuit like Field Programmable Gate Arrays (*FPGAs*) as they basically are just a string of inverters.

In [?], Sunar et al. presented a *PRNG* using stochastic jitter by combining several *ROs*. They required a post processing of the bit stream, based on resilient functions, to mask imperfections in the entropy source and to increase immunity against changes in environmental conditions. The entropy of the bit stream was used to validate the results in [?].

Wold et al. [?] proposed an enhanced version with better random characteristics and without a post processing. They only added an extra D flip-flop at each ring output. The effectiveness of their proposal was tested by means of test suites available in the open literature [?, ?, ?].

In this paper a detailed description of a very compact hardware implementation of the *RO*-based *PRNG* proposed in [?] is done. In order to validate the randomness of the noise sequences generated, two quantifiers derived from the information theory are used. They define a dual entropy plane H_{BP} vs H_{hist} . H_{hist} is a measure of the first characteristic of a *PRNG* pointed in the abstract, the equiprobability among all possible values. H_{BP} is a measure of the second characteristic pointed in the abstract, the independence between consecutive values. This methodology was successful to evaluate randomizing techniques applied to chaos-based *PRNG* [?]. A comparison with other options both physical and algorithmic, proposed in the literature is made showing that, in spite of their simplicity, *RO* are good candidates as *PRNG*.

Organization of the paper is as follows: section 4.5.3 describes the hardware implementation of the *ROs* mapped in *FPGA* Cyclone III. Section ?? shows how the normalized entropies are determined (to keep this paper short we do not detail already published results); 4.5.4 presents the results obtained for different configurations of the same *PRNGs* proposed in [?], and the statistical comparison with other utilized *PPRNGs*. Finally we present our conclusions in Sec. 5.3.

4.5.3. Implementación en Hardware

The implemented *PRNG*'s consist of several *ROs* with their outputs XORed together and sampled by a *D* flip flop, The flip flop latches the output at a

4.5. IMPLEMENTACIÓN Y ANÁLISIS ESTADÍSTICO DE TRNG BASADO EN ROS111

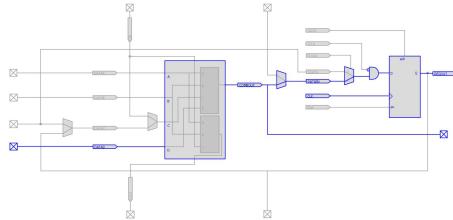


Figura 4.11: *LE* implementing an inverter and a Flip Flop, Chip Planner view.

selected frequency (here 100MHz)[?]. The physical implementation is made on *ALTERA[©] Cyclone III EP3C120* development kit with a *EP3C120F780C7N FPGA*. The design is made with *Quartus[©] II 13.1* software.

Reseña del Chip

FPGAs consist of a large number of logic array blocks (*LABs*), with groups of logic elements (*LEs*) for implementing sequential as well as combinatorial circuits. In the *Cyclone III* family architecture each *LAB* contains 16 *LEs*. Basically, each *LE* is a Flip Flop (*FF*) with a four-input look-up table (*LUT*) (see Fig. 4.11). Each *LUT* can implement any function of four variables. The *FF* and the *LUT* can be used together or independently, [?].

Usually, the logic synthesis software assigns *LE*'s resources without the designer intervention. But in the design of *RO*-based *PRNGs* it is necessary to control the exact location of each individual component for two reasons: 1) to avoid the simplification of the inverters performed by the synthesis tool; 2) to locate each *RO* in the desired place. In *Altera* the use of low-level primitives enables one to control the hardware implementation for each *cone of logic* [?]. Consequently low-level primitives and assignments are employed inside the *HDL* (hardware description language) code employed in our design.

Strings of *ROs* can be programmed on the chip by instantiating the *LUTs* as inverters. In the case of *ROs* it is necessary to prevent the *Quartus II* synthesis engine to merge two *NOT* gates in series, by using a primitive called *LCELL*. A *LCELL* always consumes one logic cell and it is not removed from the project during logic synthesis.

These primitives allow one to break up the design into manageable parts. Each cone is as small as a *LCELL* instantiation. To create a *RO*, *LCELLs* are

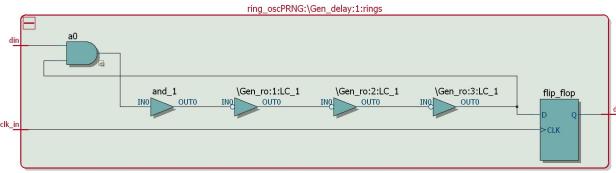


Figura 4.12: RTL view one ring with 3 inverters.

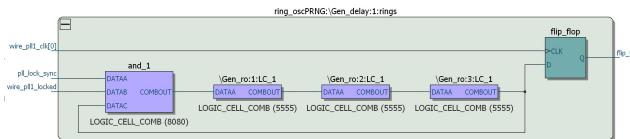


Figura 4.13: Technology map viewer (post mapping), one ring with 3 inverters.

programmed as inverter-buffers. Figs. 4.12 and 4.13 show how this primitive is implemented by the Quartus II compiler.

Furthermore, to avoid the synthesis tool to optimize removing the redundant buffers away, the Ignore *LCELL* Buffers must be set in *OFF* in the *More Analysis & Synthesis Settings* dialog box. Also *Remove Redundant Logic Cells* must be set to *OFF*.

In order to place each *RO* at a desired position, it must be assigned to a previously defined *LogicLock region*. In this way the *fitter* will keep all the elements of each ring inside the same region, [?]. The process of mapping all the elements to a particular location on the chip (*LogicLock* region) is achieved by the *Assignment Editor* tool, that also allows one to verify that the placements are actually still there, after the *Synthesis* and *Place & Route* processes.

Fig. 4.14 shows the 50 *LogicLock* regions used in this paper as they are established in the die. One *RO* is assigned to each region. Regions are spread over the die for a future analysis of location importance. Each region has 16 *LABs*, to allow us to increase the number of inverters of each ring, an issue to be considered in future work.

3-inverters, employed in a *RO* and the *FF* were all mapped onto a *LE* each, meaning that the block utilization is 4 of 16 *LEs* for any *LAB*.

Fig. 4.11 displays a single *LE*, there an inverter is implemented in the *LUT* and it can be seen the exact *LUT* input that is used. Also the output *FF* of the

4.5. IMPLEMENTACIÓN Y ANÁLISIS ESTADÍSTICO DE TRNG BASADO EN ROS113

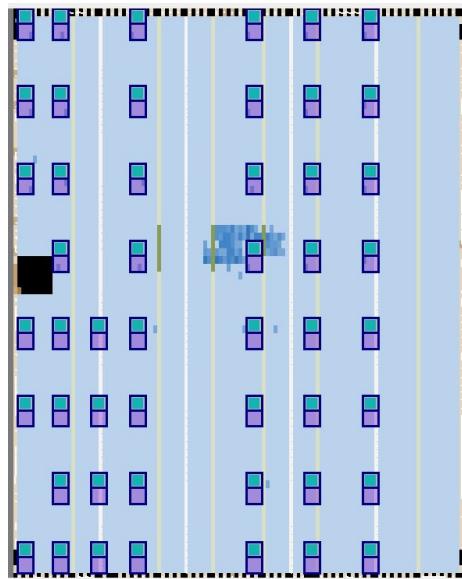
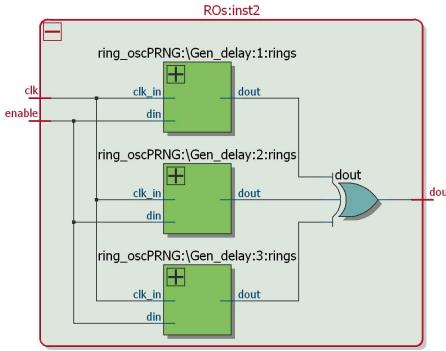


Figura 4.14: *Chip Planner* view *LogicLock* regions.

ring is mapped there.

There are many factors that determine the frequency of each *RO*, and contributes to the unpredictability of the output:

1. Placement within the *LAB*: different placements between rings could result in timing differences.
2. Connections: even having exactly identical placement of the *LUTs* with respect to each other in a given ring, it is not possible to have exactly the same *routing resource usage* in the connections. A small difference in *routing resource usage* could affect the ring delay.
3. Input selection: the *fitter* will choose which *LUT* input is utilized during the routing stage. But the delay through the *LUT* depends on which of the four inputs is used and consequently the rings could also have different delays.
4. Neighborhood: even if the design locks down all the placement and routing of a section and everything is physically locked, the timing can change by a few picoseconds depending on what is placed and routed around the ring.

Figura 4.15: *RTL view of PRNG with 3 ROs.*

Total logic elements	847/119,088	(< 1 %)
Total combinational functions	629/119,088	(< 1 %)
Dedicated logic registers	617/119,088	(< 1 %)
Total registers	617	
Total memory bits	131,072/3,981,312	(3 %)

Cuadro 4.1: Compilation Report, *RO*-based *PRNG* using 15 *ROs* and 3 inverters each.

In Fig. 4.15 (RTL view) it is shown a *PRNG* using 3 *ROs* followed by a XOR gate.

Finally, Table 4.1 shows the compilation report of the *PRNG* using 15 *ROs* each with 3 inverters.

In this paper we adopt plane H_{BP} vs H_{hist} [?] to represent each *PRNG*. A higher value in any of the entropies, H_{BP} and H_{hist} , implies an increase in the uniformity of the involved PDF's. The point (1, 1) represents the ideal point for a *PRNG* with uniform histogram and uniform distribution of ordering patterns.

4.5.4. Resultados

The *Embedded Logic Analyzer* tool is utilized for collecting the random sequences generated. It constitutes a *system-level debugging tool*, provided by *Altera* [?], that captures and stores the real-time signal behavior and allows one to observe interactions between hardware and software in system designs. After acquiring the data and save them into a *SignalTap II* file, they can be analyzed or viewed as a waveform. With this procedure nor extra jitter neither

4.5. IMPLEMENTACIÓN Y ANÁLISIS ESTADÍSTICO DE TRNG BASADO EN ROS115

distortion are introduced in the measured signal from the data acquisition chain.

In the case of *RO* based *PRNG* data files with 917504 bits each were generated for each *RO* based *PRNG*. We consider sets of N_{RO} rings, each with 3 inverters; $N_{RO} = 2, 3, 4, 5, 6, 7, 15, 25$ and 50.

Data from *SignalTap* were processed using *Matlab*[©]. Binary data were grouped in 6-bits words without superposition, so files with 152917 data each were generated. Quantifiers described in section ?? were calculated for all generated files.

We also evaluated other known noises generators to compare their quality with that of the *RO*-based *PRNG*. The noises analyzed are:

- Mersenne Twister pseudo-random number generator, [?].
- Two algorithms employed for generate random data by Matlab (Multiplicative Congruential method) [?] and Excel [?].
- Two *physical noises*: radioactive decay noise [?] and atmospheric noise [?].

Data files for these noises are available from the referred *websites*.

- Two chaotic map M^1 and their iterated versions M^2 to M^8 [?] for the logistic map (*LOGISTIC*) and the *three way Bernoulli map (TWBM)*.

Fig. 4.16 shows the results in the dual entropy plane H_{BP} vs H_{hist} for all these noises. It can be seen that the *physical noises*, the algorithmic *Mersenne Twister*, and the *PRNGs* used in *Matlab* [©](*rand* function) and *Excel*[©] (*RAND* function), have the maximum value for H_{BP} , indicating that all the ordering patterns appear almost the same number of times. However these five noises present very different behavior with respect the H_{hist} quantifier. The *radioactive decay* is the worst, with a value of H_{hist} about 0,5 indicating that this sequence does not exhibit all possible values in the same proportion. In Fig. 4.16 the numbers next to each marker for the chaotic sequences, indicate the number of iteration. The iterated maps have higher H_{BP} because of their mixing property [?].

In the case of the *RO*-based *PRNG* sequences, numbers next to each square indicate the quantity of *ROs* employed in that *PRNG* (let us stress that the number of inverters is fixed to 3). The dual entropy plane shows that an increase in the number of *ROs* improves both H_{BP} and H_{hist} .

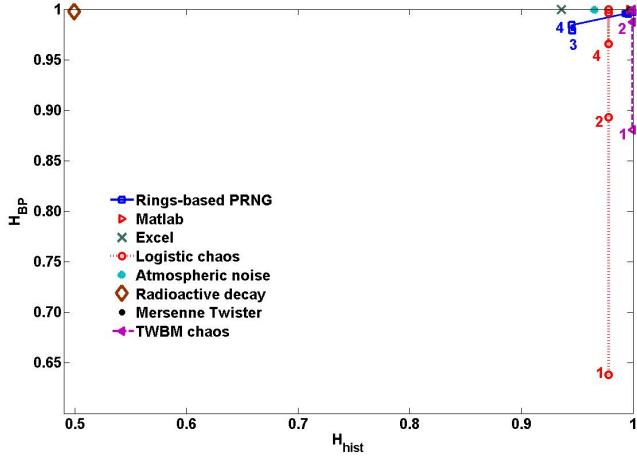


Figura 4.16: H_{BP} vs H_{hist} plane for several noises, numbers next to each square indicate the quantity of ROs used in the RO based $PRNG$. Numbers next to each point in the chaotic sequences labeled *Logistic* and *TWBM* indicate the number of iteration of the chaotic map (see the text for details).

Fig. 4.17 is a zoom of Fig. 4.16 around the ideal point (1, 1).

There, it is shown the evolution of the RO -based $PRNG$ sequences when the quantity of ROs increases from 5 to 50 (numbers next to each square). It can be seen that as the number of rings increases, data increase their mixture and also the histogram tends to be more uniform. So both properties are improved. Here a threshold in the number of rings can be determined, as the points saturate at about (0,997, 1), so this is the best $PRNG$ possible. Further, using more than 15 ROs presents no improvement. As it was previously said, H_{hist} quantifier detects the histogram variation of the sequence, and the H_{BP} quantifier reflects the improvement in the mixing of data. Finally, Mersenne Twister and Matlab sequences present identical value, ideal H_{BP} , and a high value of H_{hist} nonetheless the histogram is not perfectly uniform (values are not equiprobable).

4.6. Conclusiones

RO -based $PRNG$ implemented here has demonstrated to satisfactorily meet the statistical properties desired by a $PRNG$. They are comparable of other used

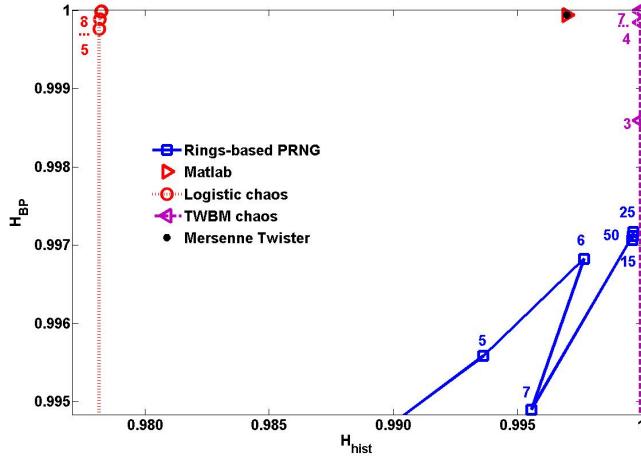


Figura 4.17: Zoom of Fig. 4.16 around the ideal point $(1, 1)$ of the H_{BP} vs H_{hist} plane. Numbers next to each square indicate the number of ROs used in that rings-based $PRNG$. Numbers next to each point in the chaotic sequences indicate the number of iteration of the chaotic map.

PRNGs and in some cases they are better. They employs few resources of the device and they are simply to implement in a digital platform.

It was demonstrated that for these architectures of $PRNG$ the quantity of ROs establishes $PRNG$'s statistical properties. It was seen that for 15 ROs both output's statistical properties, histogram and mixing, were almost ideal, making unnecessary the increase of the number of rings.

The dual entropy plane proposed here has demonstrated to satisfactorily discern between the $PRNG$'s two main desired properties, the equiprobability among all possible values and the statistical independence between consecutive values. Thus, it allows to clearly see what needs to be improved in a given sequence.

Capítulo 5

Complejidad de mapas conmutados en precisión finita

5.1. Introduction

En los últimos años, los sistemas digitales se convirtieron en el estándar en todas las ciencias experimentales. Mediante el uso de nuevos dispositivos electrónicos programables como DSP y electrónica reconfigurable como FPGA o ASIC, los experimentadores pueden diseñar y modificar sus propios generadores de señales, sistemas de medición, modelos de simulación, etc.

Cuando se implementa un sistema caótico en computadoras o cualquier dispositivo digital, el atractor caótico se vuelve periódico por el efecto de la precisión finita, entonces solo se pueden generar atractores pseudocaóticos [?, ?]. La discretización puede incluso destruir el comportamiento pseudocaótico y, en consecuencia, es un proceso no trivial [?, ?, ?].

En estos nuevos dispositivos, el punto flotante y el punto fijo son la aritmética disponible. El punto flotante es la solución más precisa, pero no siempre se recomienda cuando se requieren velocidad, baja potencia y/o área de circuito pequeño, una solución de punto fijo es mejor en estos casos.

El efecto de la discretización numérica sobre un mapa caótico fue abordado recientemente en [?] y [?]. En [?], el autor caracteriza las interfaces de Moire en el sistema Mandelbrot. Estas interfaces son consecuencia del tipo de precisión de los datos y no aparecen en el sistema ideal con números reales. En [?], los autores exploran la degradación estadística del espacio de fases para una familia de mapas cuadráticos 2D. Estos mapas presentan una dinámica multiatractor que los hace muy atractivos como generadores de números aleatorios en campos como criptografía, codificación, etc.

Grebogi y colaboradores [?] estudiaron este tema y vieron que el período T escala con el redondeo ϵ como $T \sim \epsilon^{-d/2}$ donde d es la dimensión de correlación del atractor caótico. Conseguir un período grande T es una propiedad importante de los mapas caóticos, en [?] Nagaraj *et. al* se estudió el efecto de cambiar las longitudes de período promedio de los mapas caóticos en precisión finita. Vieron que el período T del mapa compuesto obtenido al conmutar entre dos mapas caóticos es más alto que el período de cada mapa. Liu *et. al* [?] estudió diferentes reglas de conmutación aplicadas a sistemas lineales para generar caos. El problema de la conmutación también se trató en [?], el autor consideró algunos aspectos matemáticos, físicos y de ingeniería relacionados con sistemas singulares, principalmente de conmutación. Los sistemas conmutados surgen naturalmente en la electrónica de potencia y en muchas otras áreas de la electrónica digital.

La estocasticidad y la mezcla también son relevantes para caracterizar un comportamiento caótico. Para investigar estas propiedades, se estudiaron varios cuantificadores [?]. La entropía y la complejidad de la teoría de la información se aplicaron para dar una medida de la entropía causal y no causal y la complejidad causal.

Una cuestión fundamental es el criterio para seleccionar la función de distribución de probabilidad (PDF) asignada a las series de tiempo, son posibles las opciones causales y no causales. Aquí consideramos la PDF tradicional no causal obtenido al normalizar el histograma de la serie temporal. Su cuantificador estadístico es la entropía normalizada H_{hist} que es una medida de equiprobabilidad entre todos los valores permitidos. También consideramos una PDF causal que se obtiene asignando patrones de orden a segmentos de trayectoria de longi-

tud D . Este PDF primero fue propuesto por Bandt & Pompe en [?]. La entropía correspondiente H_{BP} también fue propuesta como un cuantificador por Bandt & Pompe, en [?] los autores aplicaron la complejidad causal C_{BP} para detectar el caos. Entre ellos, merece una consideración especial el uso de una representación planar de complejidad y entropía (plano $H_{hist} \times C_{BP}$) y el plano entropía causal vs. no causal (plano $H_{BP} \times H_{hist}$) [?, ?, ?, ?, ?, ?, ?].

Recientemente, la información de amplitud se introdujo en [?] para agregar cierta inmunidad al ruido débil en un PDF causal. El nuevo esquema rastrea mejor los cambios abruptos en la señal y asigna menos complejidad a los segmentos que exhiben regularidad o están sujetos a efectos de ruido. Luego, definimos la entropía causal con contribuciones de amplitud H_{BPW} y la complejidad causal con contribuciones de amplitud C_{BPW} . Además, presentamos los planos modificados $H_{hist} \times C_{BP}$ y $H_{BP} \times H_{hist}$.

Amigó y colaboradores propusieron el número de patrones prohibidos como un cuantificador de caos [?]. Básicamente, informan la presencia de patrones prohibidos como un indicador del caos. Recientemente se demostró que el nombre de patrones prohibidos no es conveniente y fue reemplazado por patrones faltantes (MP) [?], en este trabajo los autores muestran que existen sistemas caóticos que presentan MP a partir de una cierta longitud mínima de patrones. Nuestro principal interés en MP es porque da un límite superior para los cuantificadores causales.

Siguiendo [?], en este trabajo estudiamos las características estadísticas de cinco mapas, dos mapas bien conocidos: (1) los mapas tent (TENT) y (2) logístico (LOG), y tres mapas adicionales generados a partir de ellos: (3) SWITCH, generado al comutar entre TENT y LOG; (4) EVEN, generado al omitir todos los elementos en posiciones impares de la serie temporal SWITCH y (5) ODD, generados descartando todos los elementos en posiciones pares en la serie de tiempo SWITCH. Se usan números binarios flotantes y de punto fijo, estos sistemas numéricos específicos pueden implementarse en modernos dispositivos lógicos programables.

5.2. Resultados

Se estudiaron cinco mapas pseudocaóticos: dos mapas simples y tres combinaciones de ellos. Para cada uno hemos usado números representados por coma flotante (80 bits de mantisa) y números de punto fijo con $1 \leq B \leq 53$, donde B es el número de bits que representa la parte fraccionaria. Las series de tiempo se generaron usando 100 condiciones iniciales elegidas al azar dentro de su dominio de atracción (intervalo $[0, 1]$), para cada una de estas 54 precisiones numéricas.

Los mapas estudiados son: logístico (LOG), tent (TENT), conmutación secuencial entre TENT y LOG (SWITCH) y skipping descartando los valores en las posiciones impares (EVEN) o los valores en las posiciones pares (ODD), respectivamente.

El mapa logístico es interesante porque es representativo de la gran familia de mapas cuadráticos. Su expresión es:

$$x_{n+1} = 4x_n(1 - x_n) \quad (5.1)$$

con $x_n \in \mathbb{R}$.

Efectivamente, para trabajar en una representación dada, es necesario cambiar la expresión del mapa para realizar todas las operaciones en los números de representación elegidos. Por ejemplo, en el caso de LOG, la expresión en números binarios de punto fijo es:

$$x_{n+1} = 4\epsilon \text{ floor} \left\{ \frac{x_n(1 - x_n)}{\epsilon} \right\} \quad (5.2)$$

con $\epsilon = 2^{-B}$ donde B es la cantidad de bits que representa la parte fraccionaria.

El mapa tent ha sido ampliamente estudiado en la literatura porque teóricamente tiene buenas propiedades estadísticas que pueden obtenerse analíticamente. Por ejemplo, es fácil probar que tiene un histograma uniforme y, en consecuencia, un $H_{hist} = 1$ ideal. El operador Perron-Frobenius y sus autovectores y autofunciones correspondientes se pueden obtener analíticamente para este mapa [?].

El mapa tent se representa con la ecuación:

$$x_{n+1} = \begin{cases} 2x_n & , \text{if } 0 \leq x_n \leq 1/2 \\ 2(1-x_n) & , \text{if } 1/2 < x_n \leq 1 \end{cases} \quad (5.3)$$

con $x_n \in \mathbb{R}$.

En el redondeo de números fraccionarios base-2, la ecuación (??) se convierte en:

$$x_{n+1} = \begin{cases} 2x_n & , \text{if } 0 \leq x_n \leq 1/2 \\ \epsilon \text{ floor}\left\{\frac{2 - 2x_n}{\epsilon}\right\} & , \text{if } 1/2 < x_n \leq 1 \end{cases} \quad (5.4)$$

con $\epsilon = 2^{-B}$.

En la figura 5.1 se muestran los procedimientos de commutación, skipping par y skipping impar.

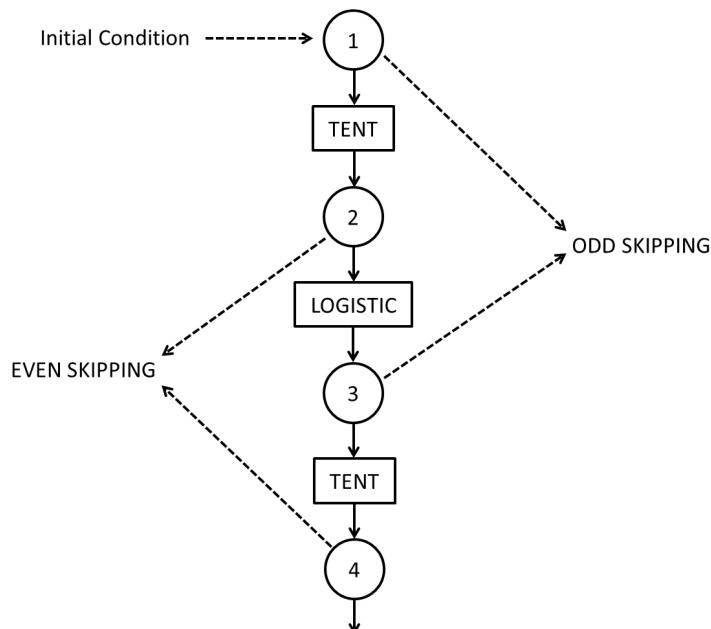


Figura 5.1: Comutación secuencial entre TENT y LOG. En la figura también se muestran las estrategias de skipping par e impar.

El mapa SWITCH se expresa como:

$$\begin{cases} x_{n+1} = \begin{cases} 2x_n, & \text{if } 0 \leq x_n \leq 1/2 \\ 2(1-x_n) & \text{if } 1/2 < x_n \leq 1 \end{cases} \\ x_{n+2} = 4x_{n+1}(1-x_{n+1}) \end{cases} \quad (5.5)$$

con $x_n \in \mathbb{R}$ y n un número par.

El skipping es una técnica habitual de aleatorización que solo aumenta la calidad de mezcla de un mapa y, por consiguiente, aumenta el valor de H_{BP} y disminuye C_{BP} de la serie temporal. El skipping no cambia los valores de H_{hist} para los mapas ergódicos porque se evalúan utilizando el PDF no causal (histograma de valores normalizado) [?].

En el caso bajo consideración, estudiamos saltos pares e impares de la conmutación secuencial de los mapas de tent y de logístico:

1. Skipping par de la conmutación secuencial de mapas Tent y Logístico (EVEN).

Si $\{x_n; n = 1, \dots, \infty\}$ es la serie de tiempo generada por la eq. ??, descarta todos los valores en posiciones impares y conserva los valores en posiciones pares.

2. Skipping impar de la conmutación secuencial de mapas Tent y Logístico.

Si $\{x_n; n = 1, \dots, \infty\}$ es la serie de tiempo generada por la eq. ??, descarta todos los valores en posiciones pares y conserva todos los valores en posiciones impares.

El skipping par se puede expresar como la función de composición TENT \circ LOG mientras que el skipping impar se puede expresar como LOG \circ TENT. La evolución del período como función de la precisión para estos mapas se informó en [?].

Los resultados para cada uno de estos mapas son los siguientes.

5.2.1. Período T en función de B

Grebogi y colaboradores [?] han estudiado cómo el período T se relaciona con la precisión. Allí vieron que el período T escala con el redondeo ϵ como

$T \sim \epsilon^{-d/2}$ donde d es la dimensión de correlación del atractor caótico.

Nagaraj *et al.* [?] estudió el caso de la conmutación entre dos mapas. Vieron que el período T del mapa compuesto obtenido al conmutar entre dos mapas caóticos es más alto que el período de cada mapa y encontraron que una conmutación aleatoria mejora los resultados. Aquí hemos considerado el solo la conmutación secuencial para evitar el uso de otra variable aleatoria, ya que esta puede introducir sus propias propiedades estadísticas en la serie temporal.

La Fig. 5.2 muestra T vs. B en escala semi logarítmica para el mapa logístico. Los puntos promediados experimentales se pueden ajustar por una línea recta expresada como $\log_2 T = mB + b$ donde m es la pendiente y b es la ordenada al origen. Los resultados para todos los mapas considerados se resumen en la tabla 5.1.

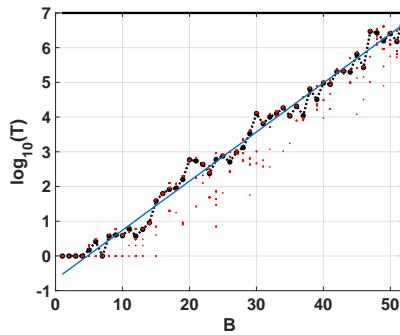


Figura 5.2: Período T en función de la preceisión B en números binarios para el mapa LOG.

Cuadro 5.1: Período T en función de la preceisión B para todos los mapas considerados

mapa	m	b
TENT	0	0
LOG	0.139	-0.6188
SWITCH	0.1462	-0.5115
EVEN	0.1447	-0.7783
ODD	0.1444	-0.7683

Los resultados son compatibles con los obtenidos en [?]. La conmutación entre mapas aumenta el período T pero el procedimiento de skipping lo disminuye casi a la mitad.

5.2.2. Cuantificadores de mapas simples

Aquí informamos nuestros resultados para mapas simples, LOG y TENT

LOG

Las Figs. 5.3a a 5.3f muestran las propiedades estadísticas del mapa LOG en representación de coma flotante y punto fijo. Todas estas figuras muestran: 100 puntos rojos (surrogados) por cada precisión de punto fijo ($1 \geq B \geq 53$) y en negro su promedio (línea negra discontinua que conecta puntos negros), 100 líneas discontinuas horizontales azules que son el resultado de cada surrogado en punto flotante y una línea continua negra en su promedio. Tenga en cuenta que estas líneas son independientes del eje x. En este caso, todas las líneas del punto flotante se superponen.

Según B crece, las propiedades estadísticas varían hasta que se estabilizan. Para $B \geq 30$, el valor de H_{hist} permanece casi idéntico al valor de la representación en coma flotante, mientras que H_{BP} y C_{BP} se estabilizan a $B > 21$. Sus valores son: $\langle H_{hist} \rangle = 0,9669$; $\langle H_{BP} \rangle = 0,6269$; $\langle C_{BP} \rangle = 0,4843$. Tenga en cuenta que el valor estable de los patrones faltantes $MP = 645$ hace que el valor óptimo sea $H_{BP} \leq \ln(75)/\ln(720) \simeq 0,65$. Entonces, $B = 30$ es la opción más conveniente para la implementación en hardware porque un aumento en el número de dígitos fraccionarios no mejora las propiedades estadísticas.

Se pueden sacar algunas conclusiones con comparando los cuantificadores de BP y BPW. Para $B = 1, 2, 3$ y 4 , los cuantificadores de BP promediados son casi 0 mientras que los cuantificadores de BPW promediados no se pueden calcular (ver en las figuras 5.3c y 5.3e la línea punteada negra faltante). Esto se debe a que para esas secuencias la condición inicial era 0, todas las iteraciones resultan ser una secuencia de ceros (el punto fijo del mapa), esto es más probable que ocurra cuando se usan pequeñas precisiones debido al redondeo.

Cuando B aumenta las condiciones iniciales se redondean a cero con menos frecuencia, esto se puede ver para $B > 6$. En este caso, las secuencias generadas que comienzan desde un valor no nulo caen a cero después de un transitorio cortomuy frecuentemente. Un tema interesante en las Figs. 5.3c y 5.3e, es que los cuantificadores de BPW muestran una alta dispersión a diferencia de los cuantificadores de BP. Esto se debe a que el procedimiento BPW tiene en cuenta

transitorios y descarta los puntos fijos, a diferencia del procedimiento BP, que considera todos los valores de la secuencia. Podemos ver en las Figs. 5.3c y 5.3e para $1 < B < 10$ líneas horizontales de puntos rojos que no aparecen en las Figs. 5.3b y 5.3d, esto evidencia que las diferentes condiciones iniciales caen en las mismas órbitas, incluso para las precisiones adyacentes.

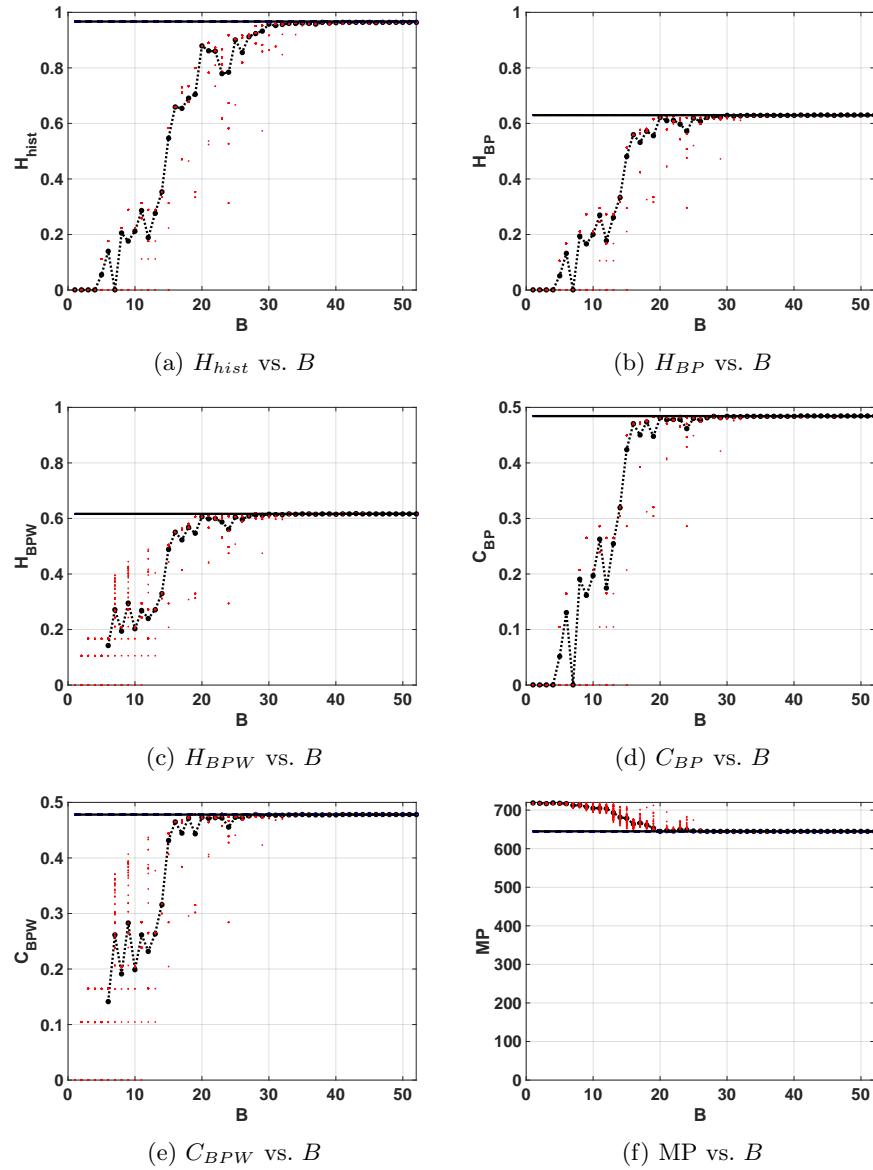
Los mismos resultados se muestran en planos de doble entropía con la precisión como parámetro (Fig. 5.4a sin contribuciones de amplitud y Fig. ?? con contribuciones de amplitud). Estas figuras muestran: 100 puntos rojos por cada precisión de punto fijo (B) y su promedio en negro (línea negra discontinua que conecta puntos negros), 100 puntos azules que son los resultados de cada surrogado en coma flotante y la estrella negra es su promedio. Aquí, los 100 puntos azules y su promedio se superponen.

Como se esperaba, la implementación de la arquitectura de punto fijo converge al valor de coma flotante a medida que aumenta B . Para ambos planos, $H_{hist} \times H_{BP}$ y $H_{hist} \times H_{BPW}$, desde $B = 20$, H_{hist} aumenta pero H_{BP} y H_{BPW} permanecen constantes. Se puede ver que la entropía de distribución de valores es alta ($\langle H_{hist} \rangle = 0,9669$) pero su mezcla es pobre ($\langle H_{BP} \rangle = 0,6269$).

En la Fig. 5.5a y 5.5b, mostramos los planos entropía - complejidad. Las líneas grises punteadas son los márgenes superior e inferior, se espera que un sistema caótico permanezca cerca del margen superior. Estos resultados caracterizan un comportamiento caótico, en el plano $H_{BP} \times C_{BP}$ podemos ver una baja entropía y alta complejidad.

TENT

Cuando este mapa se implementa en una computadora que utiliza cualquier sistema de representación numérica binaria (¡incluso punto flotante!), los errores de truncamiento aumentan rápidamente y hacen que el punto fijo inestable en $x^* = 0$ se estabilice. Las secuencias dentro del dominio de atracción de este punto fijo tendrán un corto transitorio de longitud entre 0 y B seguido de un número infinito de 0s [?, ?]. Este problema se explica de forma muy sencilla en [?], el problema aparece porque todas las iteraciones tienen una operación de desplazamiento a la izquierda que arrastra los 0s del lado derecho del número a las posiciones más significativas.

Figura 5.3: Propiedades estadísticas para el mapa LOG en función de B .

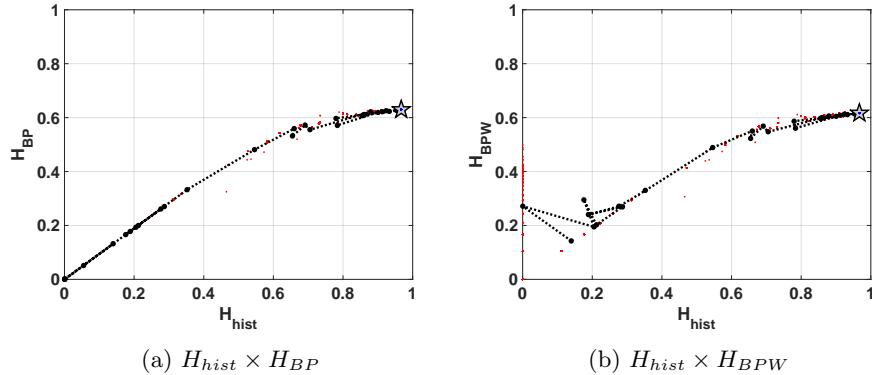


Figura 5.4: evolución de las propiedades estadísticas en el plano de doble entropía para el mapa LOG.

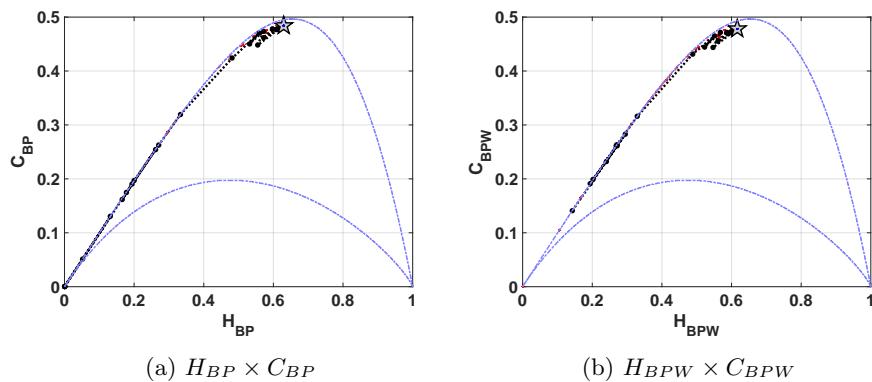


Figura 5.5: Evolution of statistical properties in causal entropy-complexity plane for LOG map

Las Figs. 5.6a a 5.6f muestran los cuantificadores para representaciones numéricas de coma flotante y fija. Los cuantificadores H_{hist} , H_{BP} y C_{BP} son iguales a cero para todas las precisiones, esto refleja que las series convergen rápidamente hacia un punto fijo para cada condición inicial. En el caso de H_{BPW} y C_{BPW} los cuantificadores son diferentes a cero porque el procedimiento BPW descarta los elementos una vez que se alcanza el punto fijo. Las altas dispersiones en H_{BPW} , C_{BPW} y MP están relacionadas con la corta duración del transitorio de la serie. Estos transitorios que convergen en un punto fijo tienen una longitud máxima de B elementos (iteraciones) para aritmética de punto fijo y 80 para punto flotante (precisión long-double).

Resumiendo, a pesar de usar un alto número de bits (con cualquier representación numérica en base 2) para representar el mapa TENT digitalizado, siempre converge rápidamente al punto fijo en $(x_n, x_{n+1}) = (0, 0)$.

5.2.3. cuantificadores de mapas combinados

Aquí presentamos nuestros resultados para las tres combinaciones de mapas simples, SWITCH, EVEN y ODD.

SWITCH

Los resultados con conmutación secuencial se muestran en las Figs. 5.7a a 5.7f. El valor de entropía calculado para la implementación en punto flotante es $\langle H_{hist} \rangle = 0,9722$, este valor es ligeramente más alto que el obtenido para el mapa LOG. Para la aritmética de punto fijo, este valor se alcanza en $B = 24$, pero se estabiliza desde $B = 28$. En cuanto a los patrones perdidos, el número de MP disminuye a 586, este valor es menor que el obtenido para el mapa LOG. Significa que la entropía H_{BP} puede aumentar hasta $\ln(134)/\ln(720) \simeq 0,74$. Los cuantificadores BP y BPW alcanzan su máximo de $\langle H_{BP} \rangle = 0,6546$ y $\langle H_{BPW} \rangle = 0,6313$ a $B = 16$, pero se estabilizan desde $B = 24$. Las complejidades son menores que para LOG, $\langle C_{BP} \rangle = 0,4580$ y $\langle C_{BPW} \rangle = 0,4578$, estos valores se alcanzan para $B \geq 15$ pero se estabilizan en $B \geq 23$. Comparado con LOG, las propiedades estadísticas son mejores con menos cantidad de bits, para $B \geq 24$ este mapa alcanza mejores características en el sentido de generador aleatorio.

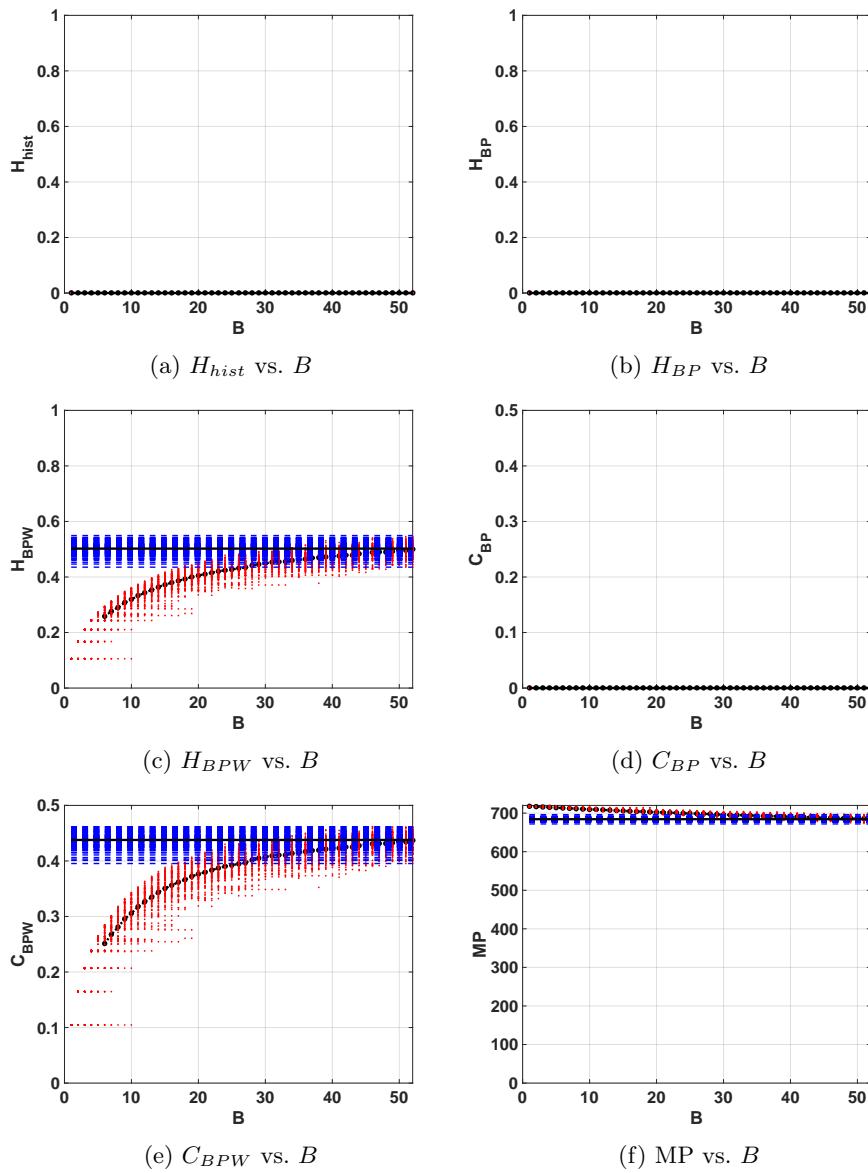


Figura 5.6: Propiedades estadísticas del mapa TENT

Además, encontramos una condición inicial con un comportamiento anómalo en doble precisión de coma flotante. Las Figs. 5.7a, 5.7b y 5.7d muestran una línea discontinua azul horizontal que está lejos del valor promedio, esto no es detectado por los cuantificadores basados en el procedimiento con contribuciones de amplitud BPW de las Figs. 5.7c y 5.7e. Sin embargo, al comparar ambos procedimientos (BP y BPW) pudimos detectar una caída a un punto fijo después de un transitorio largo, el procedimiento BPW descarta los valores constantes (que corresponden a un punto fijo) y calcula solo sobre los valores transitorios.

El plano de doble entropía $H_{hist} \times H_{BP}$ se muestra en la Fig. 5.8. El punto alcanzado en este plano para el mapa SWITCH es similar al alcanzado para el mapa LOG, y se indica con una estrella en la figura. La mezcla es ligeramente mejor en este caso.

El plano de entropía - complejidad $H_{BP} \times C_{BP}$ se muestra en la Fig. 5.9. Si comparamos con el mismo plano en el caso de LOG (Fig. 5.5a), C_{BP} es menor para SWITCH, este hecho muestra un comportamiento más aleatorio.

EVEN y ODD

En las Figs. 5.10a y 5.11a podemos ver que los cuantificadores relacionados con el histograma de valores normalizado se degradan ligeramente con el procedimiento skipping. Por ejemplo, $\langle H_{hist} \rangle$ reduce de 0,9722 sin saltarse a 0,9459 para EVEN y 0,9706 para ODD. Esta diferencia entre EVEN y ODD en coma flotante se debe a que se obtuvo una alta dispersión para H_{hist} , H_{BP} y C_{BP} pero no para H_{BPW} o C_{BPW} .

Las figuras 5.10b a 5.10f y las Figs. 5.11b a 5.11f muestran los resultados de los cuantificadores BP y BPW para EVEN y ODD, respectivamente. Se requiere una mayor precisión para lograr una complejidad menor, a diferencia de los casos sin skipping que convergen a valores altos. Desde el punto de vista de MP, se obtiene una gran mejora utilizando cualquiera de las estrategias de omisión, pero el ODD es ligeramente mejor que EVEN. Los patrones faltantes se reducen a $MP = 118$ para EVEN y ODD, lo que aumenta la entropía Bandt & Pompe máxima permitida que alcanza el valor medio $\langle H_{BP} \rangle = 0,8381$ para EVEN, y $\langle H_{BP} \rangle = 0,9094$. La complejidad se reduce a $\langle C_{BP} \rangle = 0,224$ para

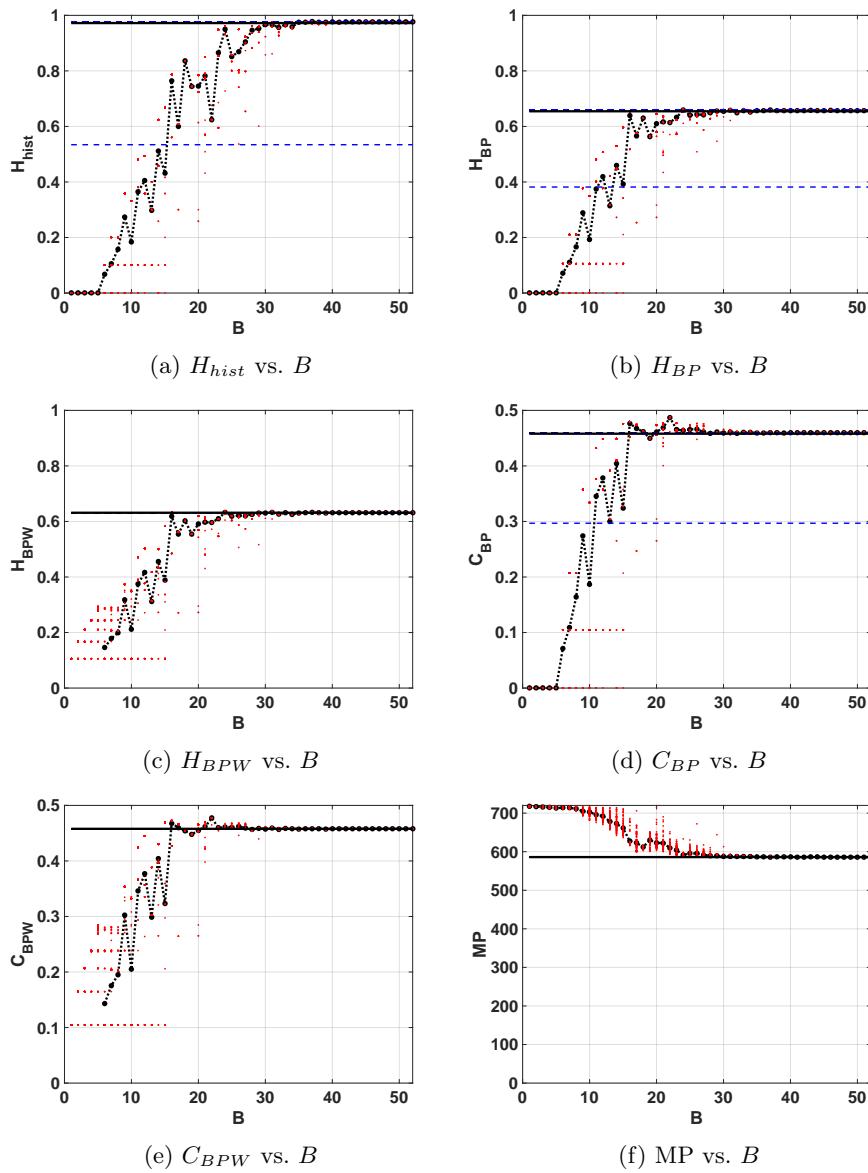


Figura 5.7: Propiedades estadísticas del mapa SWITCH

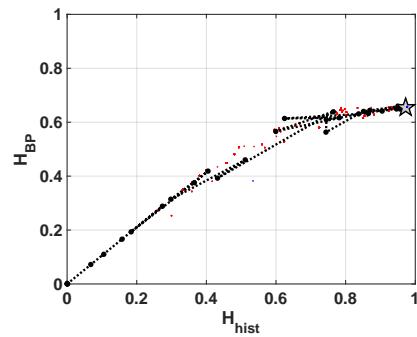


Figura 5.8: Evolución de las propiedades estadísticas en el plano doble entropía para el mapa SWITCH $H_{hist} \times H_{BP}$.

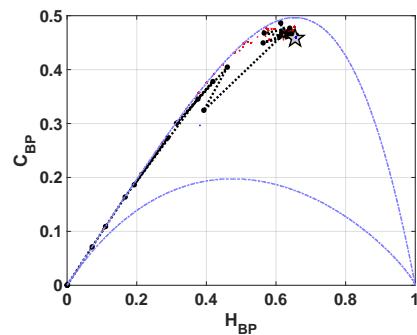


Figura 5.9: Evolución de las propiedades estadísticas en el plano entropía - complejidad para el mapa SWITCH $H_{BP} \times C_{BP}$.

EVEN y $\langle C_{BP} \rangle = 0,282$ para ODD. El número mínimo de bits para converger a este valor es de $B > 40$ para los mapas EVEN y ODD.

La mejora mostrada en las figuras 5.10 y 5.11 se refleja en la posición del punto asintótico en los planos ??, y 5.13. En ambos casos, esta posición es la más cercana al punto ideal $(H_{hist}, H_{BP}) = (1, 1)$, porque los vectores resultantes presentan una mejor mezcla.

Los resultados que se muestran en las Figs. 5.14 y 5.15 son compatibles, la posición del punto asintótico es más cercana al punto ideal $(H_{hist}, H_{BP}) = (1, 0)$. Este resultado refleja que la mezcla es mejor porque la complejidad del sistema resultante es menor. Este plano detecta que en los vectores generados por skipping, la mezcla de ODD es levemente mejor que EVEN.

Compatible results are shown in Figs. 5.14 and 5.15, the position of asymptotic point is closest to the ideal point $(H_{hist}, H_{BP}) = (1, 0)$. This result reflects that mixing is better because the complexity of resulting system is lower. This plane detects that the vector generated by ODD skipping is more mixed than EVEN.

5.3. Conclusions

Exploramos la degradación estadística debido al error inherente de los sistemas en base 2 para de mapas caóticos simples, comutados y con skipping. Evaluamos las distribuciones de mezcla y amplitud desde un punto de vista estadístico.

Este trabajo complementa los resultados anteriores dados en [?], donde se investigaron las duraciones de los períodos. En ese sentido, nuestros resultados fueron compatibles. Podemos ver que la comutación entre dos mapas aumenta la dependencia del período en función de la precisión, esto se debe a que la longitud de correlación también se incrementa. Sin embargo, el procedimiento estándar de skipping reduce la duración del período en casi la mitad.

Todas las estadísticas de los mapas representados en punto fijo producen una evolución no monótona hacia los resultados de coma flotante. Este resultado es relevante porque muestra que no siempre se recomienda aumentar la precisión.

Es especialmente interesante observar que algunos sistemas (TENT) con muy

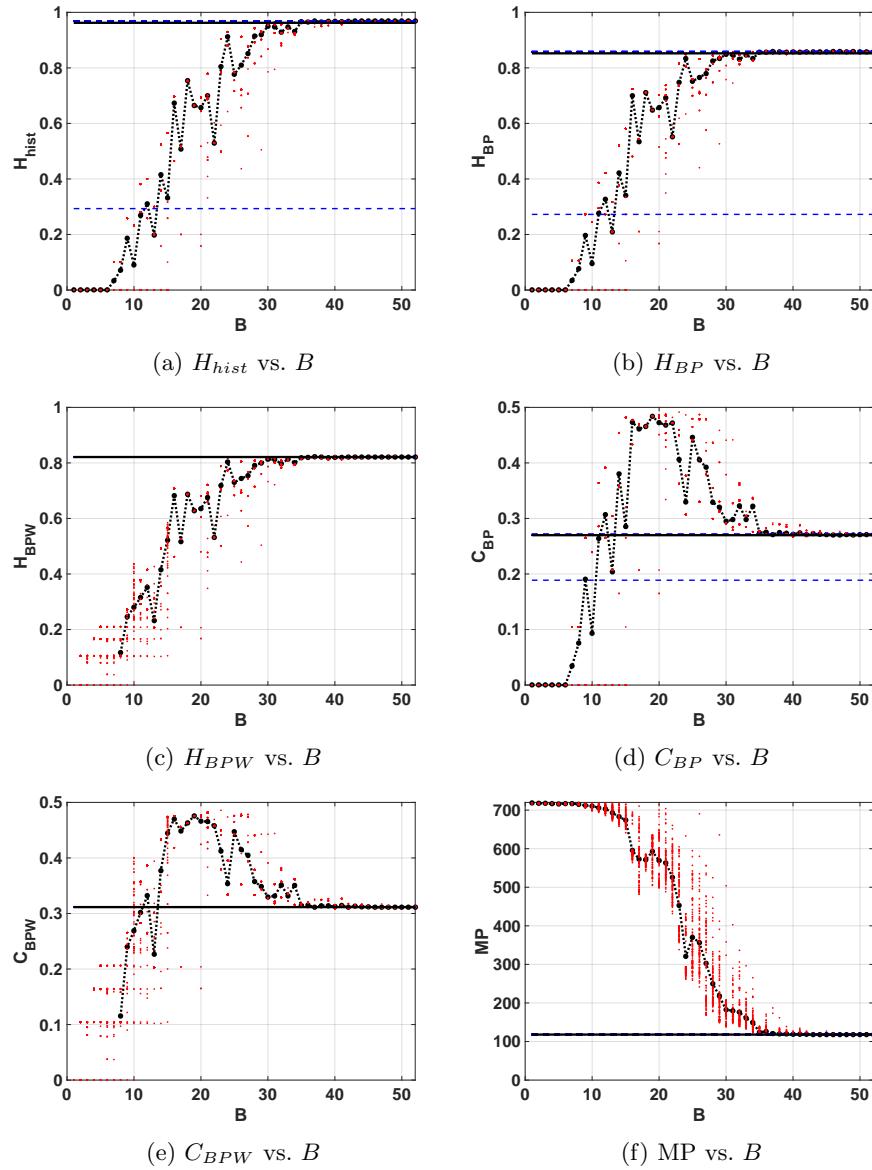


Figura 5.10: Propiedades estadísticas para el mapa EVEN

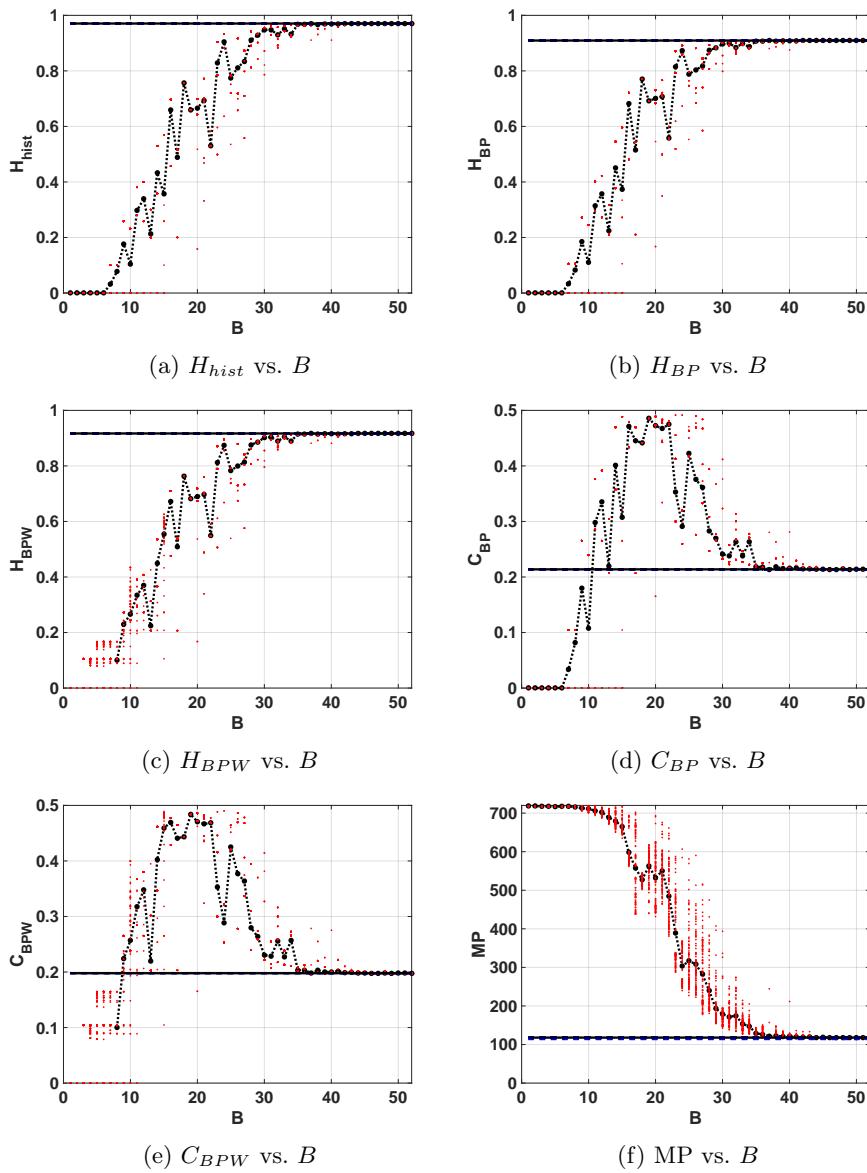


Figura 5.11: Propiedades estadísticas para el mapa ODD

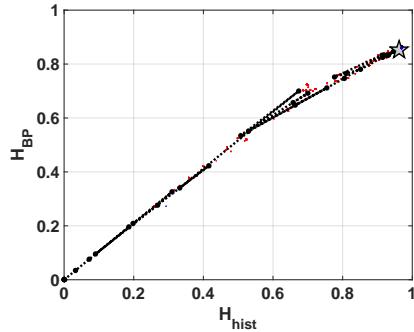


Figura 5.12: Evolución de las propiedades estadísticas en el plano doble entropía para el mapa EVEN $H_{hist} \times H_{BP}$.

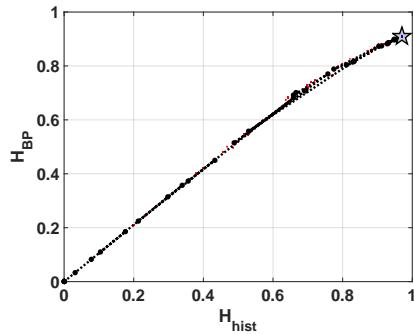


Figura 5.13: Evolución de las propiedades estadísticas en el plano doble entropía para el mapa ODD $H_{hist} \times H_{BP}$.

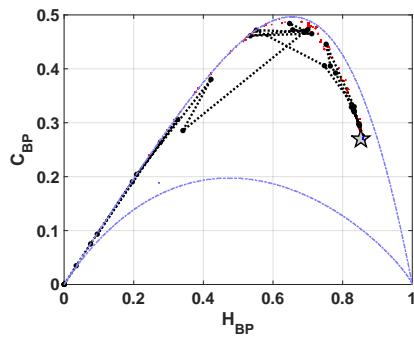


Figura 5.14: Evolución de las propiedades estadísticas en el plano entropía - complejidad para el mapa EVEN $H_{BP} \times C_{BP}$.

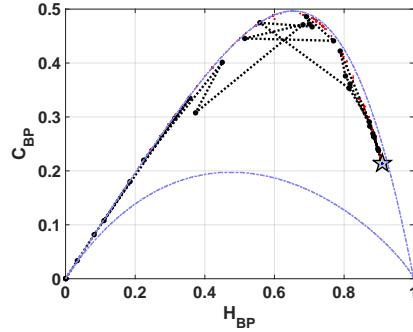


Figura 5.15: Evolución de las propiedades estadísticas en el plano entropía - complejidad para el mapa ODD $H_{BP} \times C_{BP}$.

buenas propiedades estadísticas en el mundo de los números reales, se vuelven “patológicos” cuando se usan representaciones numéricas binarias. Como regla general, si un mapa se genera solo por operaciones de shifteo (esto depende de la base de la unidad lógica aritmética y del mapa en sí), todas las condiciones iniciales convergerán a un punto fijo con un transitorio no mayor que la longitud de la mantisa que está siendo utilizada.

Al comparar los cuantificadores BP y BPW, pudimos detectar caídas a puntos fijos y pudimos estimar la longitud relativa de sus transitorios. Esto puede verse en todas las implementaciones de TENT, en una condición inicial de SWITCH y EVEN para la implementación en coma flotante.

En relación con el comportamiento estadístico, nuestros resultados muestran que SWITCH tiene una mejora marginal en la mezcla con respecto a LOG (y TENT, por supuesto). Sin embargo, la mayor mejora se produce cuando se aplica el skipping, podemos ver que las entropías de BP y BPW crecen y las complejidades de BP y BPW disminuyen, para una dada representación numérica. Este resultado es relevante porque evidencia de que un período largo no es sinónimo de buenas estadísticas, los mapas con skipping EVEN y ODD tienen longitudes de período de la mitad que los de SWITCH, pero su mezcla es mejor y sus distribuciones de amplitud se mantienen casi iguales. Como contrapartida, se necesita más precisión para alcanzar las mejores asíntotas que ofrecen el método de skipping.

Capítulo 6

Conclusiones

Apéndice A

Field Programmable Gate Array (FPGA)

Cosas que distraen en la tesis.

Bibliografía