LFSR-Fractal ANN Model Applied in R-IEDs for Smart Energy

C. Sandoval-Ruiz

***Abstract*—In this research, a novel VHDL model was developed for implementation of self-similar circuits, in reconfigurable hardware. A Reed Solomon coding system (*255,k*) was selected as a case study. Fractal design techniques and concurrent modeling ware applied. A theoretical contribution was achieved, with the logic model of a Fractal ANN. Likewise; practical advances are obtained with the optimization in terms of hardware resources and energy efficiency. The correspondence of these circuits has been interpreted through LFSR schemes, which constitutes a significant contribution for Reconfigurable IEDs applications in Reconfigurable Systems of Renewable Energy, under criteria of hardware re-usability.**

***Index Terms*—Fractal-ANN, R-IEDs, Reconfigurable Systems of Renewable Energy, LFSR, RS(*n,k*) encoder, FPGA, VHDL.**

I. Introducción

ACTUALMENTE, la migración hacia una matriz energética inteligente, requiere la optimización de los sistemas dinámicos, a través de un modelo generalizado. Por este motivo, se plantea el modelo a través de esquemas fractales, que presenten circuitos con estructuras auto-similares, en la arquitectura de los operadores básicos y bus de datos de dimensiones escalables.

Esto para aplicaciones de sistemas de potencia, control y comunicaciones, así como códigos correctores de error – CCE, para la reutilización de recursos y componentes básicos. Con capacidad de actualización sobre hardware de los módulos de gestión y comunicación de la red, convergen características SDR – *Software Defined Radio* [1-3], CR – *Radio Cognitive* [4-6], que permitirán la configuración de los componentes de la *Smart Grid*.

El objetivo de este trabajo es desarrollar un modelo configurable, para la implementación en hardware de operadores aplicables a los sistemas de control distribuido. Para tal fin se presenta una revisión de los conceptos de redes neuronales de arquitectura profunda y se identifican estructuras auto-similares en CCE, dada la complejidad de estos y su relevancia en el soporte de reconfiguración remota de la red eléctrica.

La arquitectura distribuida (unidades de procesamiento concurrente y memoria) de las ANNs – *Artificial Neural Network*, las hacen apropiadas en aplicaciones sobre hardware que soportan cómputo paralelo de funciones complejas.

Sandoval-Ruiz, C. Instituto de Matemática y Cálculo Aplicado IMYCA, Facultad de Ingeniería, Universidad de Carabobo (e-mail: cesandova@gmail.com).

Por ello, se presentan como una alternativa para generar modelos de sistemas dinámicos complejos, computo reconfigurable [7], así como CCE [8-11]. Los generadores de secuencia LFSR – *Linear Feedback Shift Register*, presentan alta coincidencia con la arquitectura de las ANN, que pueden ser diseñadas a través de estructuras concurrentes [12]. Esta característica puede ser aplicada en el modelado, estableciendo así, una correspondencia entre los componentes de los sistemas para su generalización.

A. Fractal Programmable Converters Arrays – FPCA

En el marco de los requerimientos de una infraestructura adaptativa de la red eléctrica, para actualizaciones dinámicas, reutilización de equipos, y adaptación de componentes. Se consideran los FPCA y diseños eficientes de *Hybrid Renewable Energy System* – HRES [13]. Los arreglos de convertidores de ERNC (energías renovables no convencionales), pueden ser adaptados, a través de elementos inteligentes, definiendo sistemas reconfigurables de convertidores de energías renovables – SRCE, con un arreglo con múltiples etapas de convertidores (aporte en función de la eficiencia de conversión), elementos de almacenamiento y realimentación (ver Fig. 1).

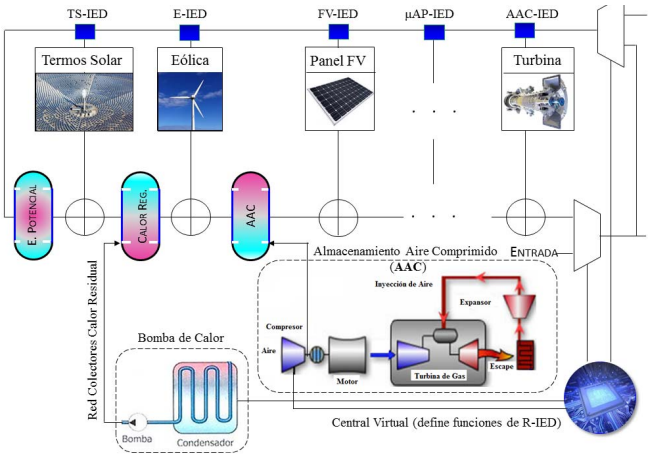

Fig. 1. Arquitectura LFSR del Arreglo de Convertidores de ERNC.

Donde se puede observar la auto-similitud entre los elementos de memoria y de almacenamiento de energía [14-16] y la relación con el modelo [17] en una configuración multi-etapa, entre sus componentes (esquema fractal).

La operación del FPCA comprende, en régimen transitorio, el aporte de energías renovables (que dependen de las condiciones climáticas) y la realimentación al sistema de energía almacenada (para re-convertirla en energía eléctrica, a través de una turbina /motor-generador), en un ciclo regenerativo.

Esto se logra, incorporando los elementos de configuración, en el sistema de control, interconectado con centrales virtuales de energía ERNC y proyectos de innovación: Smart City, micro-convertidores en diseños arquitectónicos, cometas de conversión eólica-solar, paneles solares plegables, módulos de reforestación / bio-remediación, para la protección de hábitat de especies de fauna y flora, en áreas urbanas y forestales, con criterios de responsabilidad ambiental.

B. Reconfigurable Electronic Intelligent Devices R-IEDs

Uno de los principales componentes de la red, corresponde a los módulos IEDs, en los que se define una inteligencia embebida en elementos distribuidos, con capacidad selectiva y comunicación con los componentes del sistema, a través de códigos, para la configuración remota del sistema inteligente de energía (*Smart Enegy*).

Los IEDs deben soportar funciones específicas, asociadas a los elementos convertidores (TS-IED, W-IED, PV-IED, APP-IED) de la red eléctrica distribuida FPCA. Estos módulos son diseñados, con el objetivo de monitoreo, procesamiento digital, control distribuido, funciones neuronales y comunicación con otros dispositivos del sistema.

En un IED se puede implementar: control neuronal [18], ECC [19] y operadores matemáticos. Con características de actualización en hardware (HW), procesamiento paralelo, alta capacidad de cómputo y uso con eficiente de los recursos del dispositivo. Dada la relevancia de la optimización en área/potencia, se ha seleccionado la tecnología FPGA – *Field Programmable Gate Arrays* [20], a fin de establecer un modelo reconfigurable, que cumpla con los requerimientos de diseño.

Los IEDs tendrán la capacidad de configuración de la red inteligente de energías renovables (acoplado y desacoplado de convertidores, almacenamiento de energía y realimentación selectiva), actualización de componentes, reutilización y reciclaje de materiales, con el fin de reducir el consumo de recursos. Aplicando ANN para análisis de impacto ambiental de los componentes de la red, potencialidades, condiciones climáticas, demanda de energía, capacidad, eficiencia, disponibilidad de equipos, entre otros. Todo esto demuestra la importancia de modelos orientados a hardware reconfigurable, para el soporte de los módulos componentes de los dispositivos electrónicos de la red.

C. Aplicación Códigos Reed Solomon – RS(n,k)

Un RS(*n,k*) tiene *n* símbolos en la palabra de código, *k* el número de símbolos de datos, de longitud *m* la. Este código tiene una capacidad de corrección de *(n-k)/2*. Si la trama presenta un número mayor de símbolos alterados, se aplican códigos concatenados para la detección de errores y ampliar la capacidad de corrección en códigos compuestos.

Las operaciones del código RS se realizan sobre campos finitos de *Galois* - GF, lo que tiene como ventaja resultados enteros y positivos [21]. Los multiplicadores GF [22] son implementados por un circuito LFSR, presentando una estructura fractal entre el codificador RS y sus componentes *mult_GF*, con una arquitectura circuital similar (Fig. 2). El estado del arte del código RS es presentado en [23-26].

La profundidad del modelo circuital está relacionado con la profundidad de la ANN (en función del número de capas: arreglos de neuronas que reciben datos en paralelo), o elementos fractales de la red.

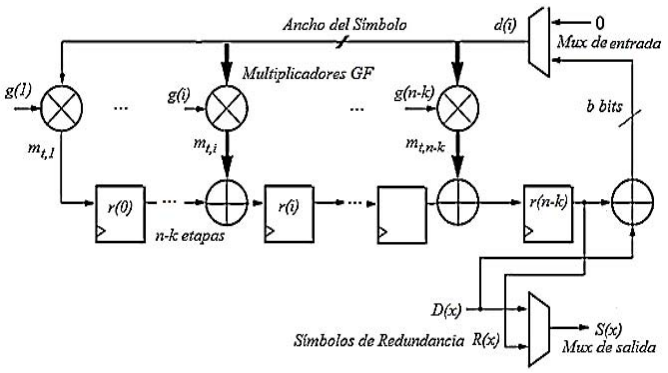

Fig. 2. Arquitectura del Codificador RS(*n,k*).

II. Redes Neuronales ANN en Hardware

En [27] se analiza la profundidad de la arquitectura de la red neuronal, ya que funciones que pueden ser representadas por *k* capas, pueden requerir un número exponencialmente grande de elementos computacionales (neuronas) para ser representadas por una arquitectura de profundidad *k-1*. Si se emplean un conjunto de subredes trabajando cooperativamente, esto equivale a añadir una capa en la arquitectura de la red de la función general, logrando solucionar problemas más complejos, que puede solucionar combinaciones de casos no entrenados.

El aprendizaje de estas redes está definido en relación a la profundidad computacional de las operaciones. El entrenamiento de DNN – *Deep Neural Network* [27], para redes supervisadas presenta demanda de capacidad de cómputo, a diferencia de redes entrenadas por medio de arquitecturas con una sola capa de neuronas ocultas [28-29]. Estas características requieren soluciones particulares para hardware [16], como es el caso del entrenamiento fraccionado, por operaciones fractales, siendo éste el enfoque seleccionado.

Los *autoencoders* [30] han sido utilizados como bloques constructores de entrenamiento de arquitecturas profundas, donde se entrena una capa independientemente de las demás, así cada capa es entrenada para codificar la entrada *x* en alguna representación correspondiente *c(x)* [27]. Se puede establecer la correspondencia con los componentes de la red fractal de codificación RS y los neuro-multiplicador GF.

Es partiendo de estos algoritmos, donde se propone un novedoso *método de entrenamiento fraccionado,* que identifica la similitud, entre el modelo compuesto por las funciones de las subredes. De esta manera, conocidos los targets de entrenamiento para las funciones componentes, se puede realizar el pre-entrenamiento supervisado de las funciones contenidas en la red neuronal fractal.

A. Fractal Artificial Neural Network – FNN

Las redes neuronales fractales pueden ser modeladas a través de funciones iteradas de los neuro-operadores, como se expresa en (1).

$$a(n) = \sum_{k=0}^{Rc} w_{j,k}\left(\sum_{i=0}^{Rf} w_{i,j}.x_i\right) \quad (1)$$

donde *a(n)* corresponde a la salida de la red, $w_{j,k}$ los pesos sinápticos de la red externa, $w_{i,j}$ los pesos sinápticos de la subred componente y x_i las entradas de la red neuronal, se tiene que la operación entre los componentes de la red están siendo operados bajo el producto de convolución, suma de productos, correspondiente a la operación de *sinapsis neuronal*.

Para el caso del codificador RS se ha considerado un modelo optimizado [24], aplicando circuitos LFCS (*Linear Feedback Concurrent Structure*), en composición fractal (Fig. 3). *Fractal ANN = LFSR_ANN (LFCS_ANN)*, definida por un arreglo de multiplicadores GF, estos operadores específicos se comportan como *auto-encoders* y el resultado es procesado en la capa de salida con estructura LFSR.

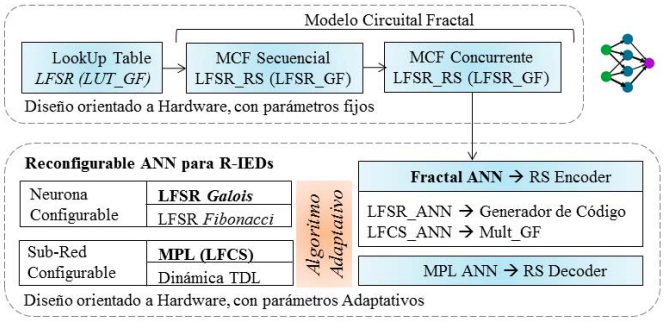

Fig. 3. Esquema Conceptual de la Fractal ANN.

Los elementos internos representan el arreglo de neuro-multiplicadores GF, y el externo corresponde al generador de secuencia RS y finalmente, estos pueden ser concatenados en codificadores 2D-RS [26].

B. Modelo de Operador Neuronal LFSR-ANN

La arquitectura de una red neuronal artificial ha sido reorganizada para la identificación de la estructura fractal (Fig.4), donde se observan los nodos, que conforman el arreglo neuronal de operadores específicos. La información procede de las señales del vector de entrada a la red *x*, procesada en la capa oculta y post-procesada en la capa de salida, a fin de obtener las señales de salida *a* de la red.

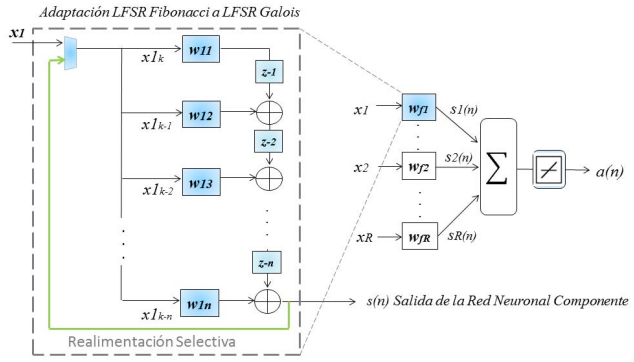

Fig. 4. Esquema General de la Red Neuronal Fractal.

El esquema permite observar, los módulos de operación neuronal, correspondientes a los elemento w_{fi}, al reordenar los componentes de la configuración del LFSR de Fibonacci a Galois [31]. Así se identifica la correspondencia del arreglo neuronal con la estructura del multiplicador GF, en relación a la arquitectura circuital auto-similar, principio que se ha discutido ampliamente en el análisis de estructuras fractales y sistemas de funciones iteradas.

Este modelo puede ser adaptado a la arquitectura fractal propia del codificador, como un arreglo compuesto entre *d(k),* entrada y la realimentación, operados con el coeficiente *g(n-k)* para cada operador LFSR(*n,k*) fractal [32], correspondiente a una muestra *k* del vector de datos, ecuación (2).

$$R(x) = \sum_{k=0}^{n} (d(k) \oplus r_{k-1}(n-k)).g(n-k) \qquad (2)$$

donde $r(x) = A(x) \bmod p(x)$, corresponde al residuo de la división entre el operando $A(x)$ de la multiplicación y el polinomio irreducible del campo finito GF(2^m). Esta operación de reducción modelar sobre el campo finito, es definida en (3).

$$a(x) = x^i A(x) \bmod p(x) \qquad (3)$$

La expresión matricial para el producto de símbolos en GF [22], puede ser expresada como en (4):

$$C(x) = \begin{bmatrix} a_{1,1} & \cdots & a_{1,m-1} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \cdots & a_{m,m-1} \end{bmatrix} \cdot \begin{bmatrix} b_{1,1} & \cdots & b_{1,m-1} \\ \vdots & \ddots & \vdots \\ b_{m,1} & \cdots & b_{m,m-1} \end{bmatrix} = [c_1 \quad \cdots \quad c_{m-1}], \qquad (4)$$

$$con\ a_{t,i} = at-1(i-1)xor\left(at-1(m-1)and\ p(i)\right)\ y\ b_{t,i} = b(i)$$

Por otra parte, la palabra de código RS [25] es generada a través de un polinomio generador *G(x)*, expresada en su forma sistemática, corresponde al bloque de información *D(x)* adicionando los símbolos de redundancia, calculados sobre el bloque de información. Este cálculo es el bloque resultante como residuo de la operación de división entre *G(x)*, operación $R_{g(x)}[.]$ aplicada sobre los símbolos de datos.

C. Modelo de SubRed Neuronal LFCS-ANN

El estudio de circuitos auto-similares LFSR en configuración *Galois*. Permiten establecer un avance en modelo concurrente LFCS, definido por variables espacio-temporales. A partir de este análisis, el circuito se ha tratado como componentes de una subred, que pueden ser configurados, según la aplicación a implementar. Estas configuraciones pueden ser adaptadas para circuitos con estructura anidada: concatenación de subredes neuronales, en arreglos concurrentes (Fig. 5).

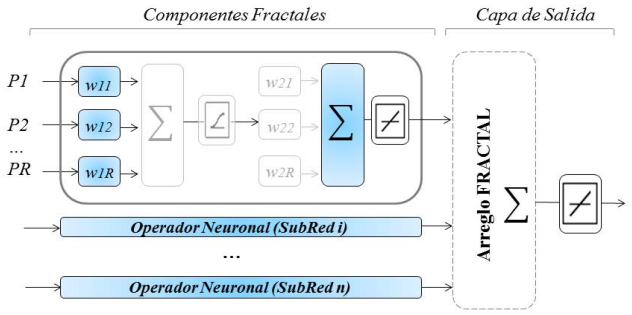

Fig. 5. Esquema Fractal ANN.

En el caso de funciones complejas con targets intermedios no conocidos, el tratamiento puede ser similar a las redes

convolucionales – *CNN*, en las que el entrenamiento de las primeras capas es no supervisado de manera de simplificar el entrenamiento entre capas de redes compuestas. Estas ANNs implementadas sobre tecnología FPGA se pueden adaptar a configuraciones específicas.

III. DESCRIPCIÓN METODOLÓGICA

Se ha seleccionado entre las funciones de los IEDs, la codificación RS, para su configuración en VHDL – *VHSIC Hardware Description Language* [24]. La descripción ha sido pensada en algebra sobre arreglos binarios. Se estudió la arquitectura y parámetros de los componentes, lo que permitió definir un esquema fractal LFSR. Se realizó la descripción del modelo, como se presentan en la Tabla 1.

TABLA I
ETAPAS DEL MODELO EN VHDL

Etapas del Modelo	Descripción de la etapa de Configuración
Definición de ANN	Se definió la arquitectura y comportamiento de los componentes de la ANN, en neuro-funciones.
Descripción en VHDL	Comprende la descripción fractal de los componentes neuronales definida en VHDL.
Pre-entrenamiento	Entrenamiento de los componentes estáticos
Configuración ANN	Descripción LFCS y LFSR
Generalización	Se diseñó el algoritmo adaptativo en VHDL.
Pruebas	Se realizó la simulación de entrenamiento.
Análisis de Eficiencia	Se analizó el rendimiento en circuito, a través de las herramientas de desarrollo.

Una vez establecida la correspondencia entre las estructuras auto-similares, se obtuvieron las ecuaciones de descripción en función de subredes, de forma modular. De esta manera, se diseñó la arquitectura del codificador como concatenación de *mult_GF*, como elementos distribuidos en subredes (5).

$$y(x) = \sum_{i=1}^{n} LFSR\_ANN(x_i) + y_{i-1} \quad (5)$$

Se adaptó a un modelo concurrente, donde se obtiene (6).

$$y(x) = \sum_{i=1}^{n} LFCS\_ANN(x_i) + y_{i-1} \quad (6)$$

Lo que permitió analizar la aplicabilidad del modelo propuesto de la fractal ANN, bajo dos enfoques. Un modelo adaptado a la arquitectura del operador específico y un modelo genérico pre-entrenado, a fin de comparar los resultados. Así mismo, se obtienen los datos de entrenamiento del *mult_GF*, para la implementación de la operación en campos finitos.

En este orden de ideas, se introduce un método de entrenamiento de aprendizaje fraccionado en componentes funcionales, lo que simplifica su implementación. Aplicando entrenamiento híbrido o aprendizaje supervisado, de forma modular, en la red neuronal de estructura fractal.

Por su parte, en la implementación del operador neuronal LFCS[12], se adaptan los pesos *w* a los coeficientes del multiplicador GF, la salida de cada neurona es calculada de forma paralela, en un compromiso entre velocidad de cómputo y área utilizada.

A. Algoritmo de Modelado en VHDL

El algoritmo de generación del código (Fig. 6), esquematiza las ecuaciones descriptivas del modelo, lo que permite crear y adaptar el código de la Fractal ANN en el circuito FPGA.

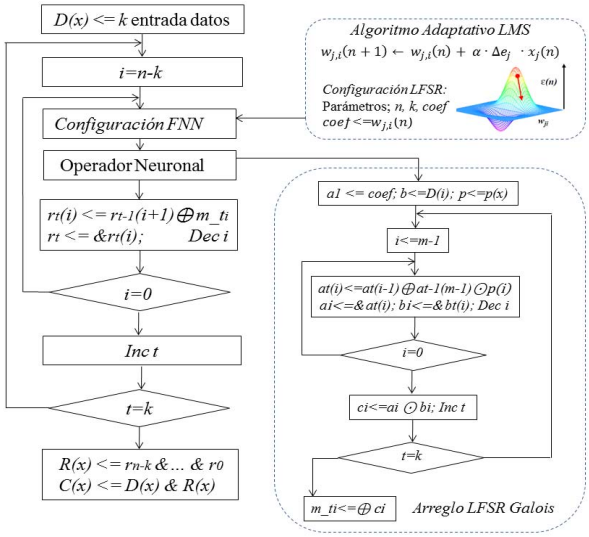

Fig. 6. Algoritmo de Generación de Código en VHDL.

Se puede observar el código generatriz para la descripción de la red fractal, el operador neuronal y un algoritmo adaptativo (propuesto) para identificación de sistemas y la configuración dinámica en hardware de la Fractal ANN. Esto hace posible su descripción a través de funciones iteradas en VHDL, para establecer una red reconfigurable, con modelo generalizado, en función de los parámetros particulares. Se seleccionó un caso de estudio para la ilustración del modelo.

B. Caso de Estudio: RS (mult_GF)

En este punto, se considera la alternativa de entrenamiento del operador neuronal *mult_GF*, fraccionado para una MPL, que permite simular una LFCS_ANN, con mayor nivel de abstracción. Se aplican los *targets* de las subredes componentes (ver Fig. 7), se describe en la arquitectura de la red neuronal del arreglo *mult_GF* VHDL, usando las ecuaciones desarrolladas.

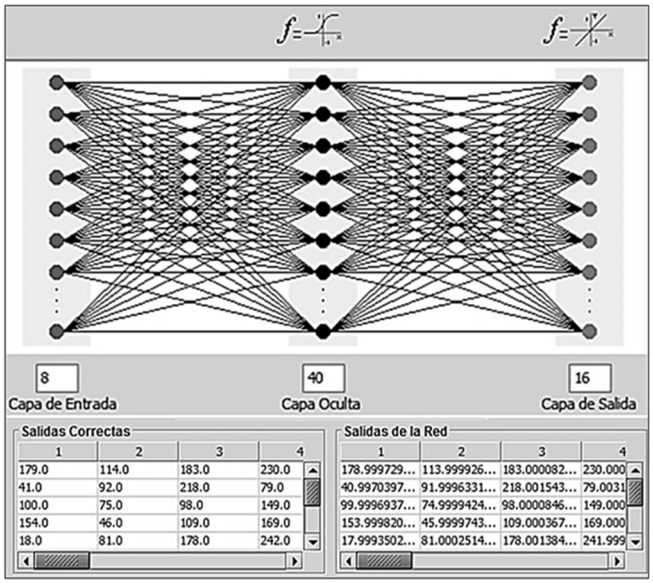

Fig. 7. Red Neuronal del Arreglo Multiplicador GF (255,247).

Se puede observar que la salida de la MPL corresponde con el *target*, la salida esperada con la que se entrenó la red neuronal. El número de neuronas, requeridas en la capa oculta ha sido de 40 neuronas. De esta manera se obtiene que el arreglo de 16 *mult_GF* del código RS(255,247) seleccionado, presenta una topología de dos capas estáticas, 8 entradas (combinación binaria del elemento a operar), 40 neuronas (en la capa oculta) multiplexadas por cada salida *k*, de manera que su implementación consume *40+k neuronas*, una línea de retardo de *n-k* elementos, que corresponde con 16 salidas (resultado de cada coeficiente), el entrenamiento reportó un error de 7.24E-7, los datos son presentados en la Tabla 2.

TABLA II
RESULTADOS DE LA RED DEL MULTIPLICADOR GF(255,247)

Topología de la Red MPL			
Puertos/Capas	Entradas	Capa Oculta	Capa de Salida
IO / Neuronas	8	40	16
Función Act.	-	Sigmoide	Lineal
Entrenamiento			
Algoritmo de Aprendizaje		Descenso del Gradiente	
Épocas del Entrenamiento		2000	
Constante de Aprendizaje		0.001	
Error de Validación		7.24E-7	

EL *mult_GF* es definida la concatenación de señales, con base al modelo LFSR *Galois* concurrente, se describe en la Tabla 3.

Así el codificador RS(*n,k*) puede ser diseñado como un modelo de redes MPL o la propuesta fractal basada en el operador neuronal LFSR adaptado. Donde cada *mult_GF* lo implementa una neurona en configuración *Galois*, con las ventajas de un tratamiento a nivel de operadores lógicos y un bus de datos para señales binarias de *m* bits, con un tratamiento *pipeline* de la red fractal. La asociación de señales en el modelo RS(*mult_GF*) ha sido descrito en la Tabla 4.

De donde se obtiene un modelo que requiere *n-k* operadores neuronales para la implementación fractal, o (*n-k)*m* para operadores LFCS (para un *clk* por símbolo, *n* por trama). En comparación con codificadores / decodificadores MPL y su algoritmo de entrenamiento [33], presentando la arquitectura neuronal de la red, en función de los parámetros *n, k* en la Tabla 5.

La adaptación de la arquitectura y coeficientes del codificador, a través de una Fractal ANN, ofrece alternativas de implementación del operador neuronal y simplificación del modelo. La neurona LFSR en topología *Galois*, con las ventajas que tiene sobre ciertas aplicaciones, la comparación entre las alternativas estudiadas (ver Tabla 6).

De esta manera se obtienen los resultados estimando el consumo de recursos expresado en neuronas. El modelo LFCS ANN presenta menor consumo de HW, menor profundidad lógica, en función de neuronas configurables y redes simples, en relación a las redes MPL. Finalmente, se establece una relación entre la aplicación diseñada y el consumo de componentes básicos de procesamiento neuronal, en función de los operadores lógicos (a través de las ecuaciones matemáticas desarrolladas en la presente investigación), como se resume en la Tabla 7.

Se puede estimar los recursos de las ANN diseñadas, en función de parámetros del código RS(*n,k*). Destacando que las Fractal ANNs presentan menor complejidad matemática, dado el menor nivel de abstracción.

TABLA III
CÓDIGO VHDL DEL MULT_GF (2^8) CONCURRENTE

```
entity mult is
port    (D_dato: in std_logic_vector (7 downto 0);
         coef: in std_logic_vector (7 downto 0);
         datox: out std_logic_vector (7 downto 0));
end mult;
architecture Behavioral of mult is
signal
b,a2,a3,a4,a5,a6,a7,a8,b1,b2,b3,b4,b5,b6,b7,b8,c1,c2,c3,c4,c5,c6,c7,c8:
std_logic_vector (7 downto 0);
signal p: std_logic_vector (8 downto 0);
begin
b<= D_dato; -- dato de entrada para la multiplicación
a1<= coef;
p<="100011101"; -- Primitive polynomial = D^8+D^4+D^3+D^2+1
(285)
u1: a2<=a1(6 downto 4)&(a1(3)xor a1(7))&(a1(2)xor a1(7))&(a1(1)xor
a1(7))&a1(0)& a1(7);
u2: a3<=a2(6 downto 4)&(a2(3)xor a2(7))&(a2(2)xor a2(7))&(a2(1)xor
a2(7))&a2(0)& a2(7);
u3: a4<=a3(6 downto 4)&(a3(3)xor a3(7))&(a3(2)xor a3(7))&(a3(1)xor
a3(7))&a3(0)& a3(7);
u4: a5<=a4(6 downto 4)&(a4(3)xor a4(7))&(a4(2)xor a4(7))&(a4(1)xor
a4(7))&a4(0)& a4(7);
u5: a6<=a5(6 downto 4)&(a5(3)xor a5(7))&(a5(2)xor a5(7))&(a5(1)xor
a5(7))&a5(0)& a5(7);
u6: a7<=a6(6 downto 4)&(a6(3)xor a6(7))&(a6(2)xor a6(7))&(a6(1)xor
a6(7))&a6(0)& a6(7);
u7: a8<=a7(6 downto 4)&(a7(3)xor a7(7))&(a7(2)xor a7(7))&(a7(1)xor
a7(7))&a7(0)& a7(7);
b1<= b(0) & b(0) & b(0) & b(0) & b(0) & b(0) & b(0) & b(0);
b2<= b(1) & b(1) & b(1) & b(1) & b(1) & b(1) & b(1) & b(1);
b3<= b(2) & b(2) & b(2) & b(2) & b(2) & b(2) & b(2) & b(2);
b4<= b(3) & b(3) & b(3) & b(3) & b(3) & b(3) & b(3) & b(3);
b5<= b(4) & b(4) & b(4) & b(4) & b(4) & b(4) & b(4) & b(4);
b6<= b(5) & b(5) & b(5) & b(5) & b(5) & b(5) & b(5) & b(5);
b7<= b(6) & b(6) & b(6) & b(6) & b(6) & b(6) & b(6) & b(6);
b8<= b(7) & b(7) & b(7) & b(7) & b(7) & b(7) & b(7) & b(7);
c1<=a1 and b1;
c2<=a2 and b2;
c3<=a3 and b3;
c4<=a4 and b4;
c5<=a5 and b5;
c6<=a6 and b6;
c7<=a7 and b7;
c8<=a8 and b8
datox<=c1 xor c2 xor c3 xor c4 xor c5 xor c6 xor c7 xor c8;
end Behavioral;
```

TABLA IV
CÓDIGO VHDL DEL RS(255,247)

```
architecture Behavioral of Codificador_RS is
--se define el tipo de variable a emplear como una matriz
type memoria is array (0 to length-1) of std_logic_vector(width-1 downto 0);
signal coef: std_logic_vector(7 downto 0);
signal dato1,dato2,dato3,dato4,dato5,dato6,dato7,dato8:
std_logic_vector(7 downto 0);
component mult is
port (D_dato: in std_logic_vector (7 downto 0);
     coef: in std_logic_vector (7 downto 0);
     datox: out std_logic_vector (7 downto 0));
end component;
port (D_dato: in std_logic_vector (7 downto 0);
       coef: in std_logic_vector (7 downto 0); -- pesos del modelo RS
      datox: out std_logic_vector (7 downto 0));
end component;
begin
C1: mult_GF port map (D_dato,"11111111",dato1);--255
```

```
C2: mult_GF  port map (D_dato,"00001011",dato2);--11
C3: mult_GF  port map (D_dato,"01010001",dato3);--81
C4: mult_GF  port map (D_dato,"00110110",dato4);--54
C5: mult_GF  port map (D_dato,"11101111",dato5);--239
C6: mult_GF  port map (D_dato,"10101101",dato6);--173
C7: mult_GF  port map (D_dato,"11001000",dato7);--200
C8: mult_GF  port map (D_dato,"00011000",dato8);--24
-- Implementación de línea de retardo TDL (red fractal dinámica RS)
process (clk)
variable memoria_v:memoria:=(others=>"00000000");
begin
  if (hab='1')  then
     D_dato<=( memoria_v(0) xor D_in);
  else
     D_dato<="00000000";
  end if;
  if (clk'event and clk='1')then
    memoria_v(0):=memoria_v(1)xor dato1;
    memoria_v(1):=memoria_v(2)xor dato2;
    memoria_v(2):=memoria_v(3)xor dato3;
    memoria_v(3):=memoria_v(4)xor dato4;
    memoria_v(4):=memoria_v(5)xor dato5;
    memoria_v(5):=memoria_v(6)xor dato6;
    memoria_v(6):=memoria_v(7)xor dato7;
    memoria_v(7):= dato8;
  end if;
```

TABLA V
RED NEURONAL DEL CODIFICADOR RS (255,K)

Topología de la Red MPL para codificador con k variable				
Puertos/Capas	k	Entradas	Capa Oculta	Capa de Salida
IO / Neuronas	247	247	8	255
IO / Neuronas	239	239	16	255
IO / Neuronas	223	223	32	255
Función de Activación		-	Sigmoide	Lineal

TABLA VI
ALTERNATIVAS DEL MODELO NEURONAL

Modelo RS(*n,k*)	Niveles	Nivel Neuro *Mult_GF*		Nivel RS
		Capa Oculta	Capa Salida	
Codificador MPL	2	*n-k*	*n*	-
Arreglo neuroGF	3	$(2^2 + 1) \cdot m$	*n-k*	1
LF-ANN Galois	2	-	*n-k*	1

TABLA VII
RESULTADOS DE LOS MODELOS NEURONALES DISEÑADOS

Aplicación	Neuronas	Entrada
Multiplicador GF(2^m)	$(2^2 + 1) \cdot m$	*m*
Codificador DNN RS(n,k)	$[(2^2 + 1) \cdot m] + (n-k)$	*k x m*
Codificador MPL RS(n,k)	*(n-k)+n*	*k x m*
Codificador F-ANN RS (n,k)	*(n-k) +1*	*k x m*

C. Implementación del Caso de Estudio sobre FPGA

El reporte de síntesis para el codificador RS(*255,k*) implementado en el dispositivo FPGA XC5VLX30, es presentado en la Tabla 8.

TABLA VIII
CONSUMO DE RECURSOS CODIFICADORES RS(255,K)

Codificadores	***k***	**Ret. (ns)**	**Frec MHz**	**LUT**	**%**	**FF**	**%**
ETSIBRAN[34]	239	-	456	203	21.6	208	38.4
G.709 [34]	239	-	483	174	8.62	181	29.2
CESR.239	239	3.89	426	159	-	128	-
CCSDS [34]	223	-	265	305	-	303	15.5
Ong. JJ [35]	223	-	-	720	57.6	415	38.3
CESR.223	223	3.81	418	305	-	256	-
CESR.247	247	3.88	429	142	-	64	

En la tabla se han contrastado los resultados del modelo desarrollado, con reportes de síntesis de IPcore [34], para los mismos parámetros de los RS(*255,k*), se alcanza una reducción de hasta 21.6% en LUTs y de 38.4 % en el consumo de elementos de memoria, respecto a la métrica de consumo de recursos del dispositivo FPGA. Así mismo, respecto a G.709 se logra un 29.2 % en reducción de elementos de memoria. Por su parte, al comparar el diseño CESR.223, con Ong JJ se tiene una reducción significativa de LUT del 57.6%. El reporte del XPower del ISE11 de potencia en (W) para los codificadores CESR desarrollados es de 0.58.

Lo que permite evidenciar que el modelo desarrollado, a través de funciones iteradas para la descripción de circuitos auto-similares, permitió implementar la aplicación RS(*255,k*) reduciendo el consumo de recursos. Destacando que se mantiene el compromiso de velocidad de procesamiento por la paralelización de los componentes internos (operadores neuronales básicos de la red fractal), definiendo así un modelo eficiente. En [36] se presenta un diseño que combina tablas y algoritmos para su implementación optimizada con recursos compartidos HW/SW [37], con altas prestaciones. Es de hacer notar que el modelo desarrollado en esta investigación está pensada para implementación de los circuitos sobre hardware, basado en unidades de cómputo neuronal. Para el caso de estudio, estas unidades son definidas por los multiplicadores GF(2^m).

D. Decodificador Neuronal

Para el neuro-decodificador (ver Fig. 8), se destaca el procesamiento paralelo por las ventajas en eficiencia asociadas a su implementación en hardware, lo que lo hace una solución neuronal alternativa [38-43].

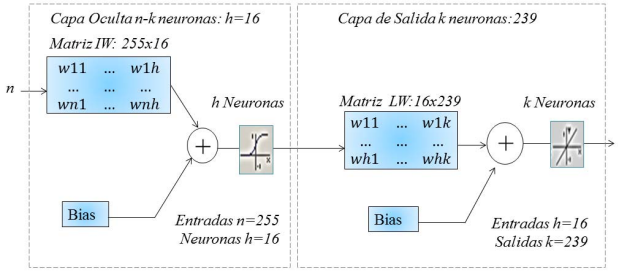

Fig. 8. Red Neuronal del Decodificador RS (255,239).

En el caso del decodificador se puede configurar una ANN, a través de una MPL *(MultiLayer Perceptron)*, con *n-k* neuronas en la capa oculta, para obtener la corrección de *(n-k)/2* errores. A la vez se ofrece una solución a la etapa del decodificador.

E. Validación del Decodificador Neuronal

Se realizaron pruebas de la red neuronal entrenada para la decodificación, donde se puede observar la corrección de errores aleatorios (ver Fig. 9).

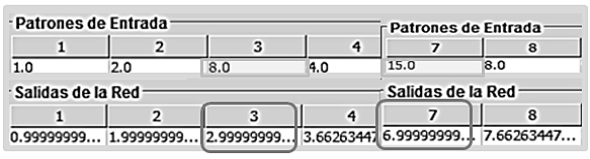

Fig. 9. Validación del Decodificador RS (255,247)

Se resume el reporte de la MPL (ver Tabla 9).

TABLA IX
RED NEURONAL DEL DECODIFICADOR RS *(255,K)*

Topología de la Red			
Arquitectura de la Red		MPL	
Puertos/Capas	Entradas	Capa Oculta	Capa de Salida
IO / Neuronas	255	*n-k*	*k*
Función Act.	-	Sigmoide	Lineal
Entrenamiento			
Algoritmo de Aprendizaje		Descenso del Gradiente	
Épocas del Entrenamiento		100	
Constante de Aprendizaje		0.001	
Error de Validación		6.55E-6	

Para cada RS(255, *k*) se definió para la capa oculta un número de neuronas correspondiente con el número de elementos de memoria *n-k*, con *k* neuronas en la capa de salida.

IV. ANÁLISIS DE RESULTADOS

Partiendo del algoritmo de generación de código, se desarrolló la descripción en VHDL del sistema. Los parámetros de configuración (pesos sinápticos) se establecen a través del entrenamiento de la red (elementos resultantes del producto en campos finitos). De esta manera, la implementación del codificador se simplifica con los parámetros de la subred pre-calculados, como entradas a la línea de retardos definida para el codificador (estructura de concatenación LFSR), que coincide con la suma ponderada de convolución propia de la red neuronal.

Los módulos son instanciados en el modelo de la Fractal ANN (ver Tabla 10). Cada uno de estos corresponde a un nivel de abstracción y dimensiones de señales definidas para la aplicación. Se realizó la generalización del código a fin de establecer las ecuaciones matemáticas descriptivas.

TABLA X
DESCRIPCIÓN VHDL DE UNA FRACTAL ANN

```
-- Componentes de la Neurona:
M1: Multiplicador port map (xi,wji,xwji);
…
Mj: Operador Neuronal port map (xi,wji,xwji); --multiplicador
S1: Sumador port map (xwji,..., b,in1);
F1: función_logsig port map (in1,out1);
-- Componentes de la Capa:
-- unidad básica de la red neuronal
-- Componentes de la SubRed Fractal
F1: FRed port map (x1,x2,...,s1);
…
FR: FRed port map (x1,c1,...,sR);
SF: Sumador port map (s1,…,sR,subred);
CL: LFSR_ANN port map (x1,c2,subred,...,cl); -- entrada externa, realimentada
end behavioral;
-- Generación de términos del LFSR:
-- ut:  at+1 <= at(i) xor (at(m-1) and p(i) & ... for i=m-1 to 0
```

En el código en VHDL, se presenta la jerarquización de los componentes, en donde se asignan los puertos de interacción entre las subredes y los elementos de procesamiento neuronal. En la definición de los elementos neuronales (ver Tabla 11), ajustando la arquitectura a un modelo LFSR generalizado, con elementos selectivos de realimentación. Así como, la opción de un algoritmo adaptativo para la actualización dinámica de los parámetros de la red.

TABLA XI
DESCRIPCIÓN VHDL DEL MODELO NEURONAL

```
-- Definición de los elementos neuronales básicos:
yp: in std_logic_vector (7 downto 0); -- definición de long. de señales
-- Definición del arreglo de registros
d1: TDL port map (in,out); -- registros de memoria del arreglo LFSR
-- Se describe la relación de la línea de retardo según la aplicación
-- LFSR_ANN: estructura de la neurona modelo LFSR propuesta
yp<= x*w1 xor…  xor x*wi …  xor x*wm  xor yp-1; --ec. general
-- relación de pesos sinápticos y las entradas al arreglo neuronal
-- Adaptación de los pesos (algoritmo adaptativo a partir de los target)
wij: alg port map (wd,xi,yi);
-- wn+1 <= función_de_aproximación (wn) –cálculo de peso actual adapt.
-- Instanciación del conjunto de operadores neuronales del arreglo FNN
nS: neurona port map (pi,wi,bi,yS); -- entrada, peso, bias, salida op. neuronal
c2: capa_lineal  port map (a1, ym); -- función de activación lineal
```

En este esquema LFSR se observa la incorporación de un componente que se ha definido como sub-red fractal, el cual se diferencia de una capa por presentar a su salida los resultados que son entrada al sumador principal de la red fractal, es decir que se obtiene una salida procesada con las características propias de la operación que define la subred, como función particular. Este es precisamente el componente que ha sido optimizado en su implementación concurrente, *LFCS_ANN*.

A. Modelado en VHDL de Funciones Iteradas – SFI

La descripción de hardware, basado en funciones iteradas, comprende la adaptación de las expresiones matemáticas que definen el comportamiento del circuito y sus componentes. En una primera etapa se realiza de una tabulación de operaciones circuitales, seguido de la descripción en VHDL, hasta obtener las ecuaciones concurrentes de síntesis (presentados en la Tabla 3), logrando la generalización del modelo (Tabla 12). Es importante destacar que este circuito realiza un conjunto de operaciones lógicas, lo que disminuye el nivel de abstracción y permite una optimización tanto en utilización de recursos del dispositivo FPGA, como en la velocidad de procesamiento (en relación a los ciclos de reloj – *clk*).

TABLA XII
DESCRIPCIÓN VHDL DEL COMPONENTE LFCS_ANN

```
entity LFCS is
port     (a: in std_logic_vector (7 downto 0);
          r: out std_logic_vector (7 downto 0));
end LFCS;
architecture Behavioral of LFCS is
signal p: std_logic_vector (8 downto 0);
begin
a<= coef;
p<="100011101";--polynomial=D^8+D^4+D^3+D^2+1(285)
-- Generación de términos del LFCS:
-- ut:at+1 <= at(i) xor (at(m-1) and p(i) & ...
u: r  <= a(6 downto 4)&(a(3)xor a(7))&(a(2)xor a(7))&(a(1)xor a(7))&a(0)&a(7);
end Behavioral;
```

El código describe el componente LFCS_ANN, al cual se hace referencia como un módulo de operación en hardware en la descripción del *mult_GF*, corresponde a la función iterativa que describe el comportamiento de la reducción modular en el campo definido por el polinomio *p(x)*.

A. Modelo Fractal ANN para VHDL

Por medio de la descripción por funciones iteradas, se logra el modelo fractal, donde para cada subred (según el nivel de profundidad circuital), tiene sus parámetros particulares: dimensión de las señales, complejidad del operador (a mayor profundidad los operadores serán circuitos auto-similares internos, capacidad de almacenamiento.

La identificación de correspondencia entre las aplicaciones estudiadas se presenta en la Tabla 13.

TABLA XIII
RELACIÓN DE CORRESPONDENCIA CIRCUITAL

LFSR / LFCS	Capas	Almacenamiento	Realimentación
Mult_GF	And Lógica	FF (1 bit)	a(t-1)
Neurona Galois	Multiplicador	Memoria	s(t-1)
RS(n,k)	Mult_GF	Reg (n bits)	r(t-1)
RS-2D	RS(n,k)	Matriz de reg.	R(t-1)
Fractal ANN	Neuro-Op.	TDL	y(t-1)
Fotovoltaica	Tándem PV	Calor Residual	λ de irradiancia
Smart Grid	IEDs- Conv.	Almac. de Energía	E. Reversible

Todos estos con la misma arquitectura definida LFSR, que define el patrón fractal. El arreglo puede ser adaptado en forma secuencial, para redes externas y circuitos concurrentes para subredes internas, de acuerdo a la aplicación.

De esta manera, se obtienen las ecuaciones generalizadas del comportamiento LFSR / LFCS se sustituyen los operadores y señales correspondientes, encontrando así un novedoso modelo para VHDL.

El modelo FNN simplifica la descripción VHDL de sistemas adaptativo, con variables acotadas. El aporte a nivel de ingeniería está dado por las ecuaciones matemáticas para generación de código hardware, como soporte para implementación de sistemas regenerativos. Esto a partir de la identificación de correspondencia con elementos de la estructura LFSR, basado en una técnica de paralelización de los componentes, reordenamiento y concatenación de operadores lógicos en un arreglo LFC(*n,k*).

A partir de la relación neuronal (1), se ha desarrollado una ecuación que describe la concatenación fractal de los operadores neuronales estandarizados en configuración *Galois*, dada por (7).

$$y(x) = \&_{i=0}^{n-k}\, y_{t\text{-}1}\,(i\text{-}1) \oplus [(x(i) \oplus y_{t\text{-}1}\,(n\text{-}k\text{-}1)) \otimes w(i)] \quad (7)$$

donde y(x) corresponde a la operación matemática LFC (*n,k*), resultante entre *x* entrada de datos y *w(x)* el polinomio característico del operador neuronal, donde se concatenan las operaciones parciales. Siendo reformulada la ecuación del operador descriptor del arreglo (8).

$$y(x) = \&_{i=0}^{n-k}\, y_{t\text{-}1}(i\text{-}1) \oplus (\oplus_{t=1}^{m} x(i) \oplus y_{t\text{-}1}(n\text{-}k\text{-}1)_t \; and \; w(i)_t) \quad (8)$$

De esta forma, los productos internos del arreglo neuronal se obtienen por elementos del campo finito GF(2^m), para cada coeficiente del polinomio generador del arreglo LFC, que coincide con los pesos característicos de la red neuronal.

El modelo teórico desarrollado permite la descripción en VHDL de los operadores basados en circuitos LFCS, estableciendo una relación directa entre operadores matemáticos y operadores neuronales. Donde, se logra definir un esquema circuital parametrizable para la estimación de recursos de hardware [44]. De esta manera, las optimizaciones realizadas sobre un circuito base pueden ser extrapoladas en aplicaciones generales.

Destacando que en el trabajo se ha manejado los niveles de abstracción, desde la descripción circuital a nivel de compuertas lógicas y concatenación de señales, para los componentes básicos (operadores neuronales), a fin de presentar avances entre el compromiso, hardware & velocidad de procesamiento. Así mismo, el algoritmo de generación de código propuesto puede ser aplicado como herramienta avanzada de síntesis en VHDL, basada en un modelo universal LFSR *Galois*.

El concepto de *LFSR_ANN* es novedoso, éste ofrece una generalización de las neuronas, con un arreglo de capas de operación paralelas, memoria y realimentación selectivas. Las estructuras neuronales han sido re-ordenadas, para coincidir con el elemento base. Donde, la optimización en la subred, que implementa las operaciones internas de la red fractal, logra una mejora exponencial en velocidad de procesamiento, al disminuir la cadena de retardos.

Se obtuvo un modelo simplificado al analizar la arquitectura similar con dimensiones escalables entre los componentes, bio-inspirado en sistema neuronal. En [45] se describen las estructuras *Recurrent Fractal Neural Networks*, lo que ha sido aplicado en ANNs. Se estudió la adaptación de los componentes y se generalizó el modelo, para optimizaciones escalables en hardware.

Los criterios de sostenibilidad del modelo se encuentras asociados a la aplicación de inteligencia colaborativa del arreglo de IEDs para mínimo consumo de energía, por ser implementadas en hardware las señales de activación representan interrupciones que no consumen recursos, ni afectan la velocidad de procesamiento.

B. Aporte en el Campo de la Ingeniería

El modelo desarrollado permitió obtener un algoritmo de síntesis aplicable en las funciones de los IEDs, para sistemas de comunicación, *Smart Grid* [46-47], *Smart Vehicle, Smart Building*, ingeniería sostenible [48], modelos regenerativos [49], entre otras. Lo que permite extrapolar el concepto tecnológico a arreglos solares reconfigurables y convertidores de energías renovables aplicando a nivel de física y mecánica cuántica, a fin de ampliar la vida útil de la tecnología en el contexto de modelo circular de hardware. Todo esto basado en criterios de sostenibilidad, respeto y responsabilidad ambiental, en las diversas escalas de los sistemas de potencia y electrónica reconfigurable. Así como diseño, formulación (por pesos adaptativos) y síntesis de materiales inteligentes, con propiedades (biodegradables) definidas por software, tele-configuración y mantenimiento regenerativo en sistemas de energías renovables. Todo esto inspirado en los procesos de la naturaleza, su equilibrio y eficiencia, con el fin de disminuir el impacto ambiental de las nuevas tecnologías.

C. Optimización Concurrente de la Subred mult_GF

En términos de eficiencia, la Fractal ANN logra un menor consumo de neuronas (en el codificador) y una simplificación del proceso de entrenamiento, lo que representa una

contribución en el diseño de IEDs, extrapolable en las funciones de soporte para la red inteligente.

La síntesis en lógica circuital, ha validado el modelo del codificador, que al compararlo con otras investigaciones [34-35] de las mismas características, para valores estándar del parámetro *k*, muestra un ahorro significativo en recursos.

D. Propuesta de Entrenamiento Parcial

Una característica del aprendizaje profundo, es la extracción de patrones de la data para su clasificación en contraste con otros métodos de aprendizaje de máquina, lo cual puede ser aplicado, para dotar al sistema de capacidad cognitiva de configuración dinámica.

Adicionalmente, se propone un algoritmo adaptativo LMS [50] en HW, para la identificación de sistema (configuración TX-RX) y adaptación de parámetros. En el cual el método de entrenamiento fraccionado aporta una solución para DNN, en las que se conocen los *targets* de las sub-redes, para definir el modelo de comportamiento modular de la subred neuronal.

V. CONCLUSIONES

Gracias al estudio de DNN y circuitos auto-similares LFSR, se desarrolló un modelo fractal, para implementación de sistemas avanzados sobre FPGA, éste simplifica el consumo de recursos y se optimiza el entrenamiento distribuido, por su tratamiento modular, lo que a su vez facilita la implementación para diseños enfocados en sistemas embebidos, como es el caso de los R-IEDs en aplicaciones de sistemas reconfigurables de convertidores de energías renovables.

Del estudio, se ha logrado establecer un algoritmo (código semilla), para la construcción del modelo, en función de las ecuaciones desarrolladas, que permiten generar el patrón fractal, donde la optimización de sub-circuitos representa un aporte en el modelo general. Por estar basado en sistemas de funciones iteradas de los componentes de la red fractal, a nivel de operadores neuronales-lógicos. Con esto se obtiene un método de implementación para Fractal ANN definidas en hardware.

Igualmente, se estableció la correspondencia entre los sistemas estudiados, lo que representa un avance para diseño de sistemas dinámicos definidos por Fractal ANN, como alternativa para modelar sistemas, de manera eficiente. La inteligencia artificial – IA ofrece alternativas de optimización en eficiencia los equipos de la red y modelos sostenibles – MS, orientado a hardware evolutivo. Aplicando técnicas de optimización dinámicas por multiplexado selectivo, desactivación de etapas (con salidas de alta impedancia), concatenación "&", paralelización (matriz de señales espacio-tiempo para modelo concurrente) y *pipeline* (segmentación de cómputo y manejo de registros de almacenamiento intermedio), basado en la naturaleza LFSR del modelo. Logrando la optimización en tiempo de procesamiento, confiabilidad, consumo de energía, tamaño y costo.

Observamos en los resultados la descripción del esquema de sub-redes neuronales, adaptadas en el modelo de un operador neuronal LFSR (configuración *Galois*), para el codificador RS(*n,k*). Donde se destaca la eficiencia del caso de estudio analizado, respecto a métricas como el consumo de recursos del FPGA, el consumo de potencia dinámica, según el reporte de síntesis y la re-utilizabilidad de recursos HW. Lo que representa un avance para diversas aplicaciones de IEDs para sistemas de potencia inteligentes, los cuales pueden ser adaptados tanto en etapa de diseño como en su implementación sobre hardware.

Siendo el principal aporte un modelo teórico de generación de código descriptor de hardware para sistemas complejos con arquitectura fractal, a través de una ecuación de generación del circuito base, con capacidad de reconfiguración del hardware de forma modular. Por otra parte, se observa la potencialidad de las Fractal ANN, para la matriz energética, que permiten la adaptabilidad, a las condiciones dinámicas de los sistemas, con alto grado de flexibilidad. La configuración fractal de la red neuronal, con funciones específicas de los módulos internos, simplifica el entrenamiento, a la vez que disminuye el tiempo de respuesta en el esquema concurrente de los circuitos, lo que las hace altamente eficientes.

REFERENCIAS

[1] D. Vázquez, "Técnicas de Codificación para control de error en Radio Definido por Software," Instituto Politécnico Nacional, 2007.

[2] V. S. Podosinov, "A Hybrid DSP and FPGA System for Software Defined Radio Applications," Polytechnic Institute and State University, 2011.

[3] C. Sandoval-Ruiz, "Módulos de Procesamiento de Señales en VHDL aplicando Redes Neuronales para Sistemas SDR," *Rev. Fac Ing UCV*, vol. 32, no. 1, pp. 17-26, 2017.

[4] A. K. Virk and A. K. Sharma, "Analysing Cognitive Radio Physical Layer on BER Performance over Rician Fading Channel," *Engineering*, vol. 2, no. 5, pp. 1697–1702, 2011.

[5] C. E. Sandoval-Ruiz, "Modelo en VHDL de Redes Neuronales Configurables Aplicadas a Decodificación en Radio Cognitivo," *Rev. Ing. UC*, vol. 24, no. 3, pp. 290-301, 2017.

[6] C. E. Sandoval-Ruiz and A. Fedón-Rovira, "Codificador RS (255,k) en hardware reconfigurable orientado a radio cognitivo," *Ing. y Univ.*, vol. 17, no. 1, pp. 77–91, 2013.

[7] Harrison, W. L., Graves, I., Procter, A., Becchi, M., & Allwein, G. (2016, June). A programming model for reconfigurable computing based in functional concurrency. In *2016 11th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)* (pp. 1-8). IEEE.

[8] J. L. Wu, Y. H. Tseng, and Y. M. Huang, "Neural Network Decoders for Linear Block Codes," *Int. J. Comput. Eng. Sci.*, vol. 3, no. 2, pp. 235–255, 2002.

[9] Shlezinger, N., Eldar, Y. C., Farsad, N., & Goldsmith, A. J. (2019, July). ViterbiNet: Symbol Detection Using a Deep Learning Based Viterbi Algorithm. In *2019 IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)* (pp. 1-5). IEEE.

[10] He, Y., Zhang, J., Wen, C. K., & Jin, S. (2019). TurboNet: A model-driven DNN decoder based on max-log-MAP algorithm for turbo code. *arXiv preprint arXiv:1905.10502*.

[11] C. Sandoval, "FPGA prototyping of neuro-adaptive decoder," *Proc. 9th WSEAS Int.*, pp. 99–104, 2010.

[12] Sandoval-Ruiz, C. Codificador RS (n, k) basado en LFCS: caso de estudio RS (7, 3). *Revista Facultad de Ingeniería*, (64), pp. 68-78, 2012.

[13] M. Pereira,E. Rego, M. Nagano, "A Multiobjective Optimization Model for the Design of Hybrid Renewable Energy Systems",*IEEE Latin America Transactions*, VOL. 16, NO. 12, pp. 2925-2933, 2018

[14] Lladó Sánchez, A. *Estudio de almacenamiento de energía mediante aire comprimido: los sistemas CAES (compressed air energy storage)* (Bachelor's thesis, Universitat Politècnica de Catalunya), 2015.

[15] Clemente Jul, M. D. C. Comparación de tecnologías de almacenamiento energético provenientes de energías renovables.

In *Anales de la Real Academia de Doctores de España*, Vol. 16, No. 1, pp. 29-49. Real Academia de Doctores de España, 2012.
[16] Guacaneme, J. A., Velasco, D., & Trujillo, C. L. Revisión de las características de sistemas de almacenamiento de energía para aplicaciones en micro redes. *Información tecnológica*, *25*(2), pp. 175-188, 2014.
[17] C. E. Sandoval-Ruiz, "Control de Micro-Redes de Energía Renovable a través de estructuras LFCS Reconfigurables en VHDL," *Cienc. y Tecnol.*, vol. 18, pp. 71-86, 2018. https://doi.org/10.18682/cyt.v1i18.847
[18] C. Sandoval-Ruiz, "Modelo VHDL de Control Neuronal sobre tecnología FPGA orientado a Aplicaciones Sostenibles," *Ingeniare Rev. Chil. Ing.*, vol. 27(3), 2019.
[19] C. Sandoval-Ruiz, "Códigos Reed Solomon para Sistemas Distribuidos de Energías Renovables y Smart Grids a través de Dispositivos Electrónicos Inteligentes sobre Tecnología FPGA". *Revista Memoria - Investigación en Ingeniería*, Vol. 16, pp. 37–54, 2018.
[20] J. J. Rodríguez Andina, E. De la Torre Arnanz, and M. D. Valdés Peña, *FPGAs Fundamentals, Advanced Features, and Applicationes in Industrial Electronics*. CRC Press, 2017.
[21] Rodríguez-Olivares, N. A., Gómez-Hernández, A., Nava-Balanzar, L., Jiménez-Hernández, H., & Soto-Cajiga, J. A. FPGA-based data storage system on NAND flash memory in RAID 6 architecture for in-line pipeline inspection gauges. *IEEE Transactions on Computers*, *67*(7), pp. 1046-1053, 2018.
[22] C. Sandoval-Ruiz, "VHDL Optimized Model of a Multiplier in Finite Fields," *Ing. y Univ.*, vol. 21, no. 2, pp. 195–211, 2017.
[23] Kato, K., & Choomchuay, S. An Analysis of Time Domain Reed Solomon Decoder with FPGA Implementation. *IEICE TRANSACTIONS on Information and Systems*, *100*(12), pp. 2953-2961, 2017.
[24] C. Sandoval-Ruiz, "Modelo Optimizado del Codificador Reed-Solomon (255,k) en VHDL a través de un LFSR paralelizado," Tesis Doctoral, Universidad de Carabobo, Venezuela, 2013.
[25] C. Sandoval-Ruiz and A. Fedón-Rovira, "Efficient RS (255 ,k) encoder over reconfigurable systems," *Rev.Téc.Ing.Zulia*, vol. 37, no. 2, pp. 151–159, 2014.
[26] C. E. Sandoval-Ruiz, "Logical-Mathematical Model of Encoder 2D-RS for Hardware Description in VHDL," *Rev. Ing. UC*, vol. 24, no. 1, pp. 28–39, 2017.
[27] E. D. De la Rosa M., "El aprendizaje profundo para la identificación de sistemas no lineales," Centro de Investigación y de estudio Avanzados del Instituto Politécnico Nacional, México, D.F., 2014.
[28] H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin, "Exploring strategies for training deep neural networks," *J. Mach. Learn. Res.*, vol. 10, pp. 1–40, 2009.
[29] D. Erhan, P.-A. Manzagol, Y. Bengio, S. Bengio, and P. Vincent, "The difficulty of training deep architectures and the effect of unsupervised pretraining," in *in Proceedings of The Twelfth International Conference on Artificial Intelligence and Statistics*, pp. 153–160, 2009.
[30] Balevi, E., & Andrews, J. G. (2019). Autoencoder-Based Error Correction Coding for One-Bit Quantization. *arXiv preprint arXiv:1909.12120*.
[31] Sandoval-Ruiz, C. Modelo de estructuras reconfigurables con registro desplazamiento, para lenguaje descriptor de hardware VHDL. *Revista de la Facultad de Ingeniería UCV*, *31*(3), 63-72, 2016.
[32] C. E. Sandoval-Ruiz, "Análisis de Circuitos Fractales y Modelado a través de Sistema de Funciones Iteradas para VHDL," *Rev. Cienc. e Ing.*, vol. 38, no. 1, pp. 3–16, 2017.
[33] C. Sandoval-Ruíz, "Modelo Neuro-Adaptativo en VHDL, basado en circuitos NLFSR, para control de un Sistema Inteligente de Tecnología Sostenible," *Universidad, Cienc. y Tecnol.*, vol. 21, no. 85, pp. 140–149, 2017.
[34] Xilinx, 2019. Performance and Resource Utilization for Reed-Solomon Encoder v9.0 LogiCORE IP Product Guide, Design Suite Release 2018.3. [Online]. Available: https://www.xilinx.com/support/documentation/ip_documentation/ru/rs-encoder.html#virtex7
[35] Ong, J. J., Ang, L. M., & Seng, K. (2011). FPGA implementation reed solomon encoder for visual sensor networks. In *2011 International Conference on Telecommunication Technology and Applications* (pp. 88-92). IACSIT Press.
[36] Almeida, Gabriel Marchesan, et al. "A Reed-Solomon algorithm for FPGA area optimization in space applications." *Second NASA/ESA Conference on Adaptive Hardware and Systems (AHS 2007)*. IEEE, 2007.
[37] G. Marchesan Almeida, "Códigos Corretores de Erros em Hardware para Sistemas de Telecomando e Telemetria em Aplicações Espaciais," Pontifícia Universidade Católica do Rio Grande do Sul, 2007.
[38] S. Coric, I. Latinovic, and A. Pavasovic, "A neural network FPGA implementation," in *in Proceedings of the 5th Seminar on Neural Network Applications in Electrical Engineering NEUREL*, 2000, pp. 117–120.
[39] Doan, N., Hashemi, S. A., & Gross, W. J. "Neural Successive Cancellation Decoding of Polar Codes". In *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)* (pp. 1-5). IEEE, 2018.
[40] Z. Salsic, "FPGA Prototyping of RNN Decoder for Convolutional Codes," *EURASIP J. Appl. Signal Processing*, pp. 1–9, 2006.
[41] W. Li, L., Yuan, B., Wang, Z., Sha, J., Pan, H., & Zheng, "Unified architecture for Reed-Solomon decoder combined with burst-error correction," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 20, no. 7, pp. 1346–1350, 2012.
[42] S. J. Su, Y. S., Yu, C., Lin, B. S., Cheng, P. H., & Chen, "Design of a (255, 239) Reed-Solomon decoder using a simplified step-by-step algorithm," in *In Consumer Electronics (ISCE), 2013 IEEE 17th International Symposium on(pp.). IEEE.*, 2013, pp. 247–248.
[43] Y. S. No Yu, C., & Su, "Two-Mode Reed–Solomon Decoder Using A Simplified Step-by-Step Algorithm.," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 62, no. 11, pp. 1093–1097, 2015.
[44] Sandoval-Ruiz C.E. "Métodos Numéricos en Diferencias Finitas para la Estimación de Recursos de Hardware FPGA en arquitecturas LFSR(n,k) Fractales". *Ingeniería, Investigación y Tecnología*, vol. XX, no. 3, pp.1-10, 2019. http://dx.doi.org/10.22201/fi.25940732e.2019.20n3.032
[45] Bieberich, E. Recurrent fractal neural networks: a strategy for the exchange of local and global information processing in the brain. *Biosystems*, *66*(3), 145-164. 2002.
[46] Qiu, R. C., Hu, Z., Chen, Z., Guo, N., Ranganathan, R., Hou, S., & Zheng, G. (2011). Cognitive radio network for the smart grid: Experimental system architecture, control algorithms, security, and microgrid testbed. *IEEE Transactions on Smart Grid*, *2*(4), 724-740.
[47] Muralitharan, K., Sakthivel, R., & Vishnuvarthan, R. (2018). Neural network based optimization approach for energy demand prediction in smart grid. *Neurocomputing*, *273*, 199-208.
[48] Steckler, D. J., Nava, C., Duarte, J., Zambrano, J., & Sandoval-Ruiz, C. E. (2018). Design of Neural Networks on microcontrollers, applied in functional modules for an eco-park. *Rev. Ing. UC*, *25*(1), 50-60.
[49] C. Sandoval-Ruiz, "Plataforma de Investigación de Redes Eléctricas Reconfigurables de Energías Renovables aplicando Modelos LFSR", *Revista Universidad, Ciencia y Tecnología* (UCT), Vol. 23, no. 95, 2019.
[50] Castellanos Hernández, J. A., Sandoval Ruiz, C. E., & Azpúrua Auyanet, M. A. (2014). A FPGA implementation of a LMS adaptative algorithm for smart antenna arrays. *Revista Técnica de la Facultad de Ingeniería Universidad del Zulia*, *37*(3), 270-278.

Sandoval-Ruiz, Cecilia. Electrical Engineer in 2002, mention Systems and Automation, Master in Electrical Engineering in 2007, and PhD in Engineering in 2014, graduated from the University of Carabobo. She has been a Titular Professor in the Postgraduate of Engineering at the University of Carabobo, as well as coordinator of the Research Group on Applied Digital Technologies. She has written more than 50 scientific articles, her areas of research are Reconfigurable Hardware, Design with Sustainable Technologies, Programmable Control Systems, Neural Networks, FPGA devices, Collaborative Research and Systems Modeling in VHDL language. Dr. C. Sandoval-Ruiz has been accredited as Researcher level C of the PEII Research and Innovation Stimulus Program.